Engineering Safety- and Security-Related Requirements for Software-Intensive Systems

Full Day Tutorial

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Donald Firesmith
31 May 2007

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
31 MAY 2007		**00-00-2007 to 00-00-2007**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Engineering Safety- and Security-Related Requirements for Software-Intensive Systems	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University ,Software Engineering Institute (SEI),Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**190**	

Tutorial Goals

Familiarize Requirements, Safety, and Security Engineers with:

- *Common Concepts and Terminology underlying each other's Disciplines*
- *Useful Reusable Techniques from each other's Disciplines*
- Different Types of *Safety- and Security-related Requirements*
- Common Consistent Collaborative *Method* for Engineering these Requirements

Enable Requirements, Safety, and Security Teams to Collaborate Together to better Engineer Safety- and Security-Related Requirements

Decrease the incidence of Accidents and Successful Attacks due to Poor Safety- and Security-Related Requirements

Contents

Three Disciplines:

Requirements, Safety, and Security Engineering

Safety and Security Engineering

Safety Engineering

> the engineering discipline within systems engineering concerned with lowering the risk of *unintentional unauthorized* harm to valuable assets to a level that is acceptable to the system's stakeholders by preventing, detecting, and reacting to accidental harm, mishaps (i.e., accidents and incidents), hazards, and safety risks

Security Engineering

> the engineering discipline within systems engineering concerned with lowering the risk of *intentional unauthorized* harm to valuable assets to a level that is acceptable to the system's stakeholders by preventing, detecting, and reacting to malicious harm, misuses (i.e., attacks and incidents), threats, and security risks

Major Differences (Safety and Security):

- Unintentional vs. Intentional
- Accidental vs. Malicious Harm
- Mishaps vs. Misuses
- Hazards vs. Threats

Requirements Engineering

Requirements Engineering

> the engineering discipline within systems/software engineering consisting of the cohesive collection of all *tasks* that are primarily performed to produce the *requirements* and *other related requirements work products* for an *endeavor*

This includes safety- and security-related requirements

Challenges:

Combining Requirements, Safety, and Security Engineering

You Are Here

Three Disciplines

Challenges _

Requirements Engineering Overview

Safety and Security Engineering Overview

Types of Safety- and Security-related Requirements

Common Consistent Collaborative Method

Conclusion

Challenges$_1$

Requirements Engineering, Safety Engineering, and Security Engineering:

- Different *Communities*
- Different *Disciplines* with different Training, Books, Journals, and Conferences
- Different *Professions* with different *Job Titles*
- Different fundamental underlying *Concepts* and *Terminologies*
- Different *Tasks, Techniques,* and *Tools*

Safety and Security Engineering are:

- Typically treated as *Secondary Specialty Engineering* Disciplines
- Performed Separately from, largely Independently of, and Lagging Behind the primary Engineering Workflow
  (Requirements, Architecture, Design, Implementation, Integration, Testing)

Challenges$_2$

Current separate Requirements, Safety, and Security Methods are Inefficient and Ineffective.

Separation of Requirements Engineering, Safety Engineering, and Security Engineering:

- Causes *poor* Safety- and Security-related Requirements that are often:
  - Goals rather than Requirements
  - Vague, unverifiable, unfeasible, architectural and design constraints
  - Inadequate and too Late to drive Architecture and Testing
- Unnecessarily harder to achieve Certification and Accreditation

Challenges$_3$

Poor Requirements are a *Primary Cause* of more than half of all Project Failures (defined in terms of):

- Major Cost Overruns
- Major Schedule Overruns
- Major Functionality Not Delivered
- Cancelled Projects
- Delivered Systems that are Never Used

Poor Requirements are a major *Root Cause* of many (or most) Accidents involving Software-Intensive Systems.

Most Security 'Requirements' mandated tend to be:

- Industry Best Practices
- Security Functions or Subfunctions

Challenges$_4$

How Safe and Secure is Safe and Secure *enough*?

Situation Cries out for Process Improvement:

- Better Consistency between Safety and Security Engineering
  - More consistent Concepts and Terminology
  - Reuse of Techniques across Disciplines
  - Less Unnecessary Overlap and Avoidance of Redundant Work
- Better Collaboration:
  - Between Safety and Security Engineering
  - With Requirements Engineering
- Better Safety- and Security-related Requirements

Requirements Engineering:
An Overview

You Are Here

Three Disciplines

Challenges

Requirements Engineering Overview _

Safety and Security Engineering Overview

Types of Safety- and Security-related Requirements

Common Consistent Collaborative Method

Conclusion

Requirements Engineering Topics

Definition of Requirements Engineering

Requirements Engineering:

- Tasks
- Work Products

Importance and Difficulty of Requirements Engineering

Goals vs. Use Case Scenarios vs. Requirements

Types of Requirements

Characteristics of Good Requirements

Definition

Requirements Engineering

> the engineering discipline within systems/software engineering consisting of the cohesive collection of all *tasks* that are primarily performed to produce the *requirements* and *other related requirements work products* for an *endeavor*

Development/Life Cycle

Not Waterfall

Requirements engineering is typically performed in an *iterative*, *recursively incremental*, *parallel,* and *time-boxed* manner:

- Iteratively:
  - Repeated as defects are found and corrected
- Recursively Incremental:
  - Subsystem by subsystem
  - From tier to tier (typically top-down)
  - From block/milestone to block/milestone
- Parallel means concurrently with the:
  - Subsystem by subsystem
  - Primary work flow disciplines such as architecting, design, and testing
  - Specialty engineering disciplines such as safety and security engineering
- Time-Boxed:
  - Milestone and Inch Pebble based
  - Avoid Analysis Paralysis

Requirements Engineering Tasks

Business Analysis (i.e., Customer, Competitor, Market, Technology, and User Analysis as well as Stakeholder Identification and Profiling)

Visioning

Requirements Identification (a.k.a., Elicitation)

Requirements Reuse

Requirements Prototyping

Requirements Analysis

Requirements Specification

Requirements Management

Requirements Validation

Scope Management (Management)

Change Control (Configuration Management)

Quality Control (Quality Engineering)

Requirements Engineering Work Products

Business Analyses

Stakeholder Profiles

Vision Statement

- **Goals**

Operational Concept Document (OCD)

- **Use Cases and Usage Scenarios**

Requirements Repository and published Specifications

- **Requirements**

Requirements Prototypes

Domain Model

Glossary

Difficulty of Requirements Engineering

"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later."

F. Brooks, *No Silver Bullet*, IEEE Computer, 1987

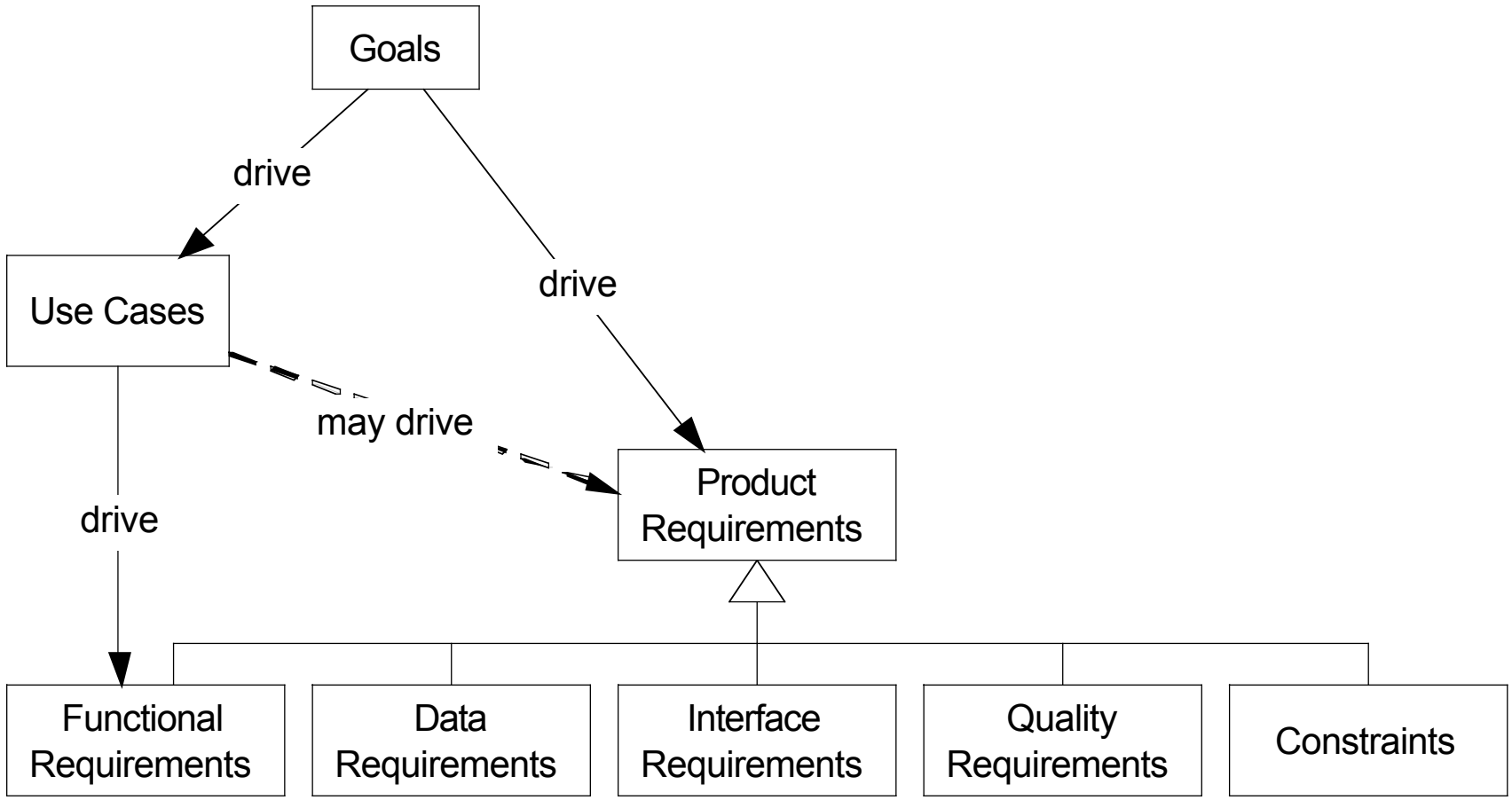
Importance and Difficulty of Requirements Eng.

Poor Requirements are a Primary Cause of more than half of all:

- Project Failures (defined in terms of):
  - Major Cost Overruns
  - Major Schedule Overruns
  - Major Functionality Not Delivered
  - Cancelled Projects
  - Delivered systems that are Never Used
- Hazards and associated Mishaps (Accidents and Safety Incidents)
- Vulnerabilities

The extent of the impact of Poor Requirements on Threats and associated Misuses (Attacks and Security Incidents) is much less clear.

Goals vs. Usage Scenarios vs. Requirements



Goals

Goal

an *Informally Documented Perceived Need* of a *Legitimate Stakeholder*

Goals are:

- *Not* Requirements.
- Drive the Identification and Analysis of the Requirements.
- Typically Ambiguous and/or Unrealistic (i.e. Impossible to Guarantee).

Major Problems with Safety and Security Goals:

- Ambiguous
- Not100% Feasible
- Not Verifiable

Typically documented in a Vision Statement.

Example Goals

The ATM will enable account holders to withdraw funds from their accounts.

The ATM will be very easy to use.

The ATM will be secure from cyber-attack.

Use Case, Use Case Path, and Usage Scenario

Usage Scenario

a *Specific* Functionally Cohesive Sequence of Interactions between User(s), the System, and potentially other actors that Provides Value to a Stakeholder

Use Case

a Functionally Cohesive Class of Usage Scenarios

Use Case Path

an Equivalence Set of Usage Scenarios that follow the Same Course through a Use Case

Use Case, Use Case Path, and Usage Scenario

Use Case Paths:

- Can be either “Sunny Day / Happy Path” or “Rainy Day”
- Should have Preconditions, Triggers, and Postconditions
- Often Documented with Sequence Diagrams or Activity Diagrams

Use Cases, Use Case Paths, and Usage Scenarios:

- Typically documented in an Operational Concept Document (OCD)
- Drive the Identification and Analysis of the [primarily functional] requirements
- Often include potential Design Information
- Can be documented using Diagrams, Lists, and/or Paragraphs

Use Cases are more than just Use Case Diagrams

Requirements

(Product) **Requirement**

> a *Mandatory* Characteristic (behavior or attribute) of a Product (e.g., System, Subsystem, Software Application, or Component)

Requirements are:

- Documented in Requirements Specifications
- Stored and Managed in Requirements Repositories / Tools

Requirements are Driven by Goals.

Example:

> "At each taxi station while under normal operating conditions, the system shall provide a taxi to passengers within an average of 5 minutes of the passengers' request."

Requirements should have certain Characteristics (e.g., verifiable and feasible).

Characteristics of Good Requirements

Mandatory

Correct

Cohesive

Feasible

Relevant

Unique

Unambiguous

Validatable

Verifiable

What or How Well, not How

Complete

Consistent

Usable by Stakeholders

Uniquely Identified

Traced

Externally Observable

Stakeholder-Centric

Properly Specified

Prioritized

Scheduled

Managed

Controlled

http://www.jot.fm/issues/issue_2003_07/column7

Some Problems due to Poor Requirements

Ambiguous Requirements:

- Developers misinterpret Subject Matter Expert (SME) intentions.
- "The system shall be safe."
- How safe? Safe in what way?

Incomplete Requirements:

- Developers must guess SME intentions.
- "The system shall do X."
- Under what conditions? When in what state? When triggered by what event? How often? How fast? For whom? With what result?

More Potential Problems

Missing Requirements:

- What shall the system do if it can't do X?
  - Numerous *rainy day* scenarios.
  - Detection? Reaction?
- Hazards are unusual Combinations of Conditions that cause Accidents.
- What shall the system do if event X occurs when the system is simultaneously in states Y and Z?

Unnecessary Constraints:

- Inappropriate Architecture and Design Constraints *unnecessarily* specified as Requirements such as:
  - User ID and Password for Identification and Authentication.

Poor Requirements Cause Accidents$_1$

"Software-related accidents are usually caused by flawed requirements. Incomplete or wrong assumptions about the operation of the controlled system can cause software related accidents, as can incomplete or wrong assumptions about the required operation of the computer. Frequently, omitted requirements leave unhandled controlled-system states and environmental conditions."

Nancy G. Leveson, 2003

<http://www.safeware-eng.com/index.php/white-papers/accidents>

Poor Requirements Cause Accidents$_2$

Large percentage of Accidents are caused by Poor Requirements:

- "For the 34 (safety) incidents analyzed, 44% had inadequate specification as their primary cause."

  Health and Safety Executive (HSE), *Out of Control: Why Control Systems Go Wrong and How to Prevent Failure* (2nd Edition), 1995

- "The majority of software-related accidents are caused by requirements errors."

  Nancy G. Leveson, 2003

- "Almost all accidents related to software components in the past 20 years can be traced to flaws in the requirements specifications, such as unhandled cases."

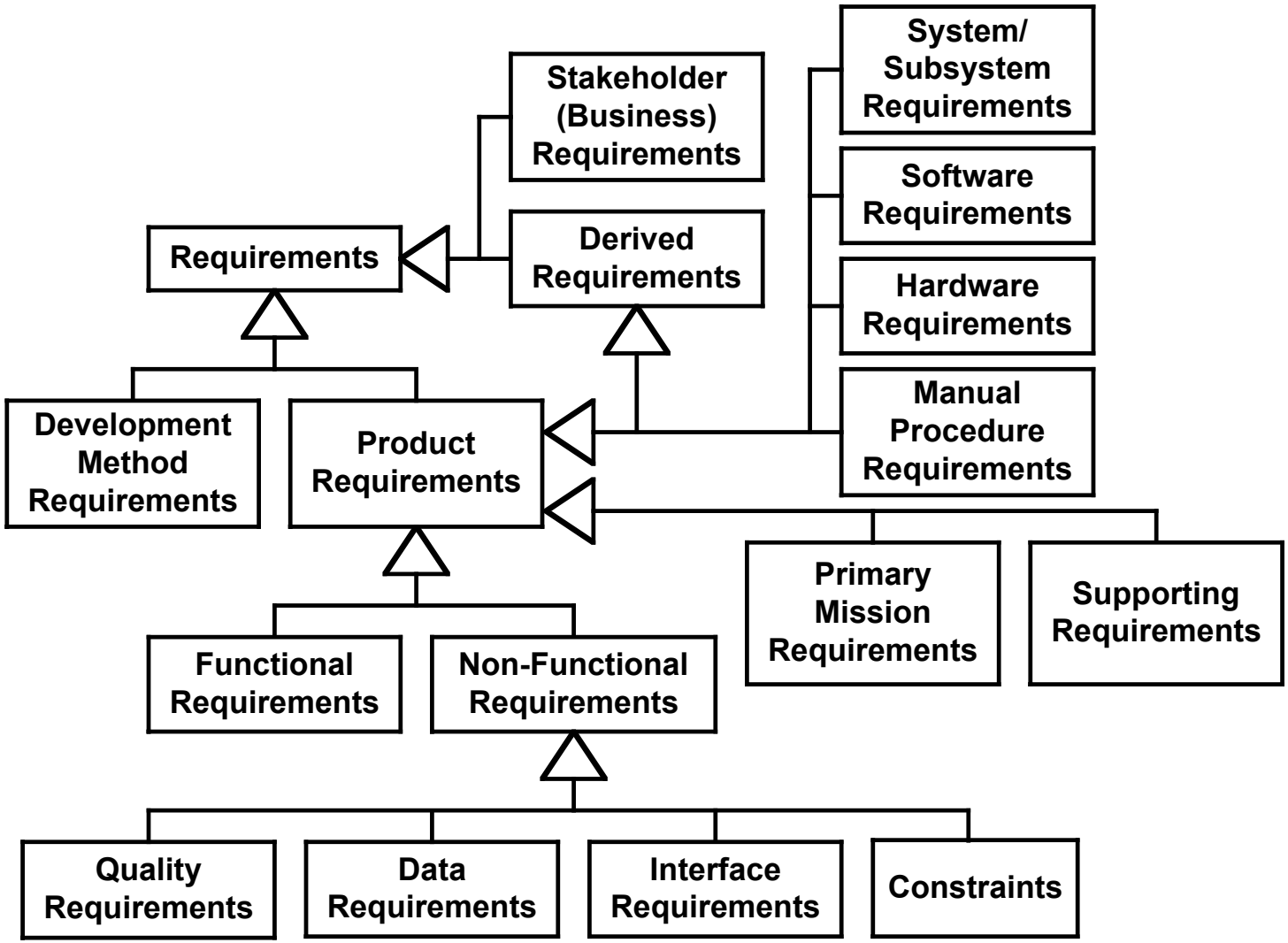
  Safeware Engineering, "Safety-Critical Requirements Specification and Analysis using SpecTRM"*, 2002*

Poor Requirements Cause Accidents$_3$

Erroneous specification is a major source of defects and subsequent failure of safety-critical systems. Many failures occur in systems using software that is perfect, it is just not the software that is needed because the specification is defective."

John C. McKnight, "Software Challenges in Aviation Systems, 2002

Types of Requirements



Product Requirements

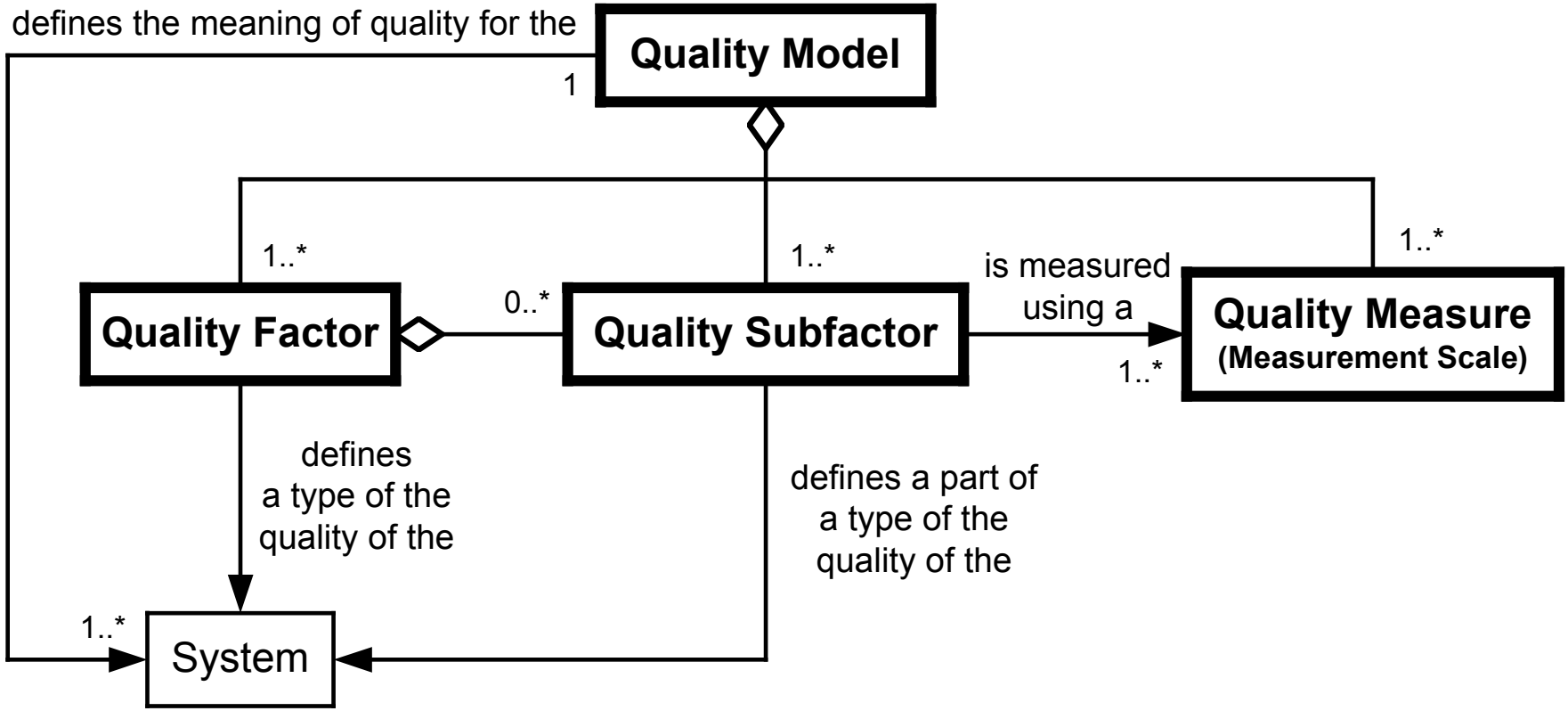
A **product requirement** is a requirement for a *product* (e.g., system, subsystem, software application, or component).

- A **functional requirement** is a product requirement than specifies a mandatory *function* (i.e., behavior) of the product.
- A **data requirement** is a product requirement that specifies mandatory [types of] data that must be manipulated by the product.
- An **interface requirement** is a product requirement that specifies a mandatory interface with (or within) the product.
- A **quality requirement** is a product requirement that specifies a mandatory amount of a type of product quality.
- A **constraint** is a property of the product (e.g., design decision) that would ordinarily not be a requirement but which is being mandated as if it were a normal requirement

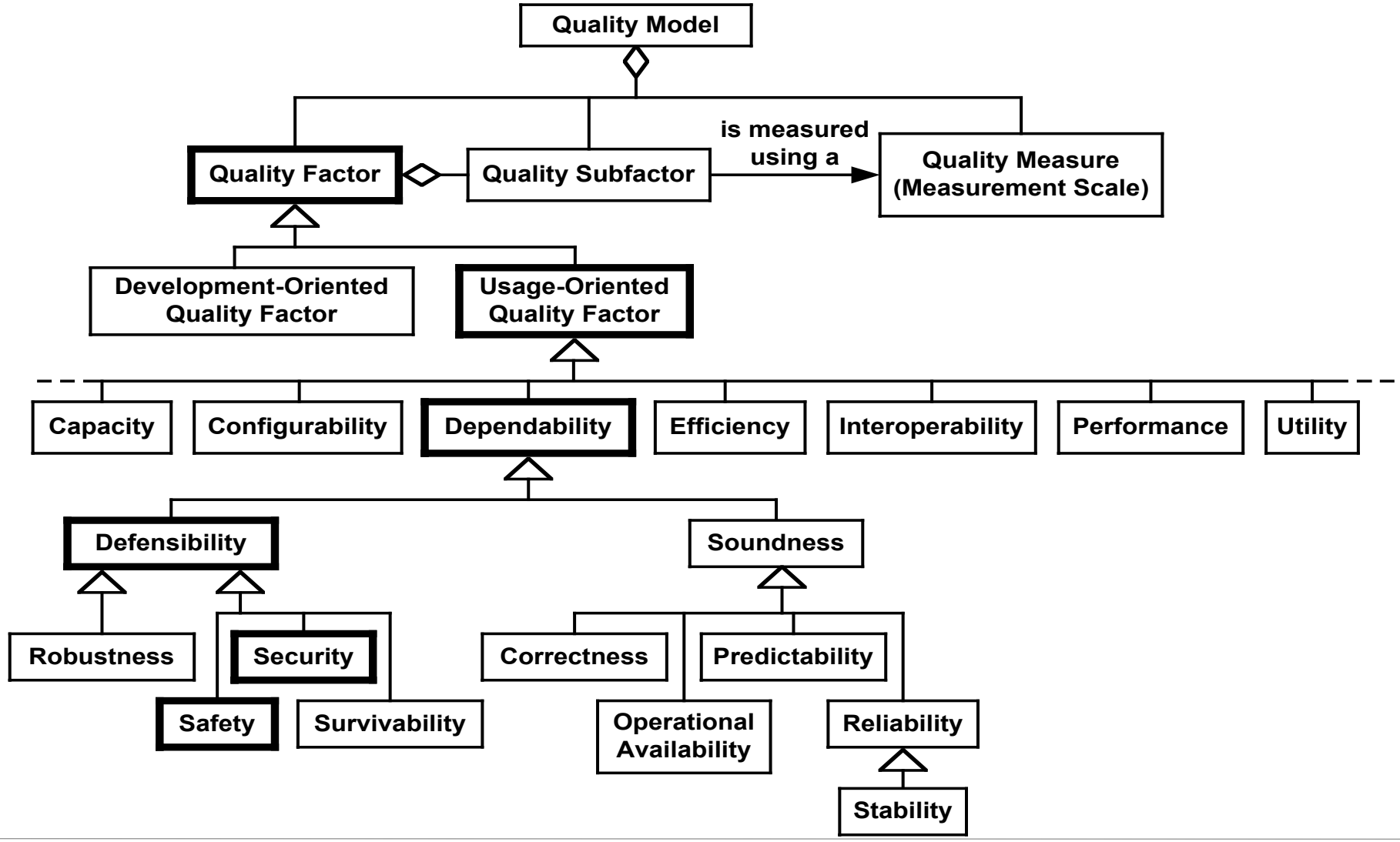
Quality Requirements:

Specify Minimum Acceptable Quality

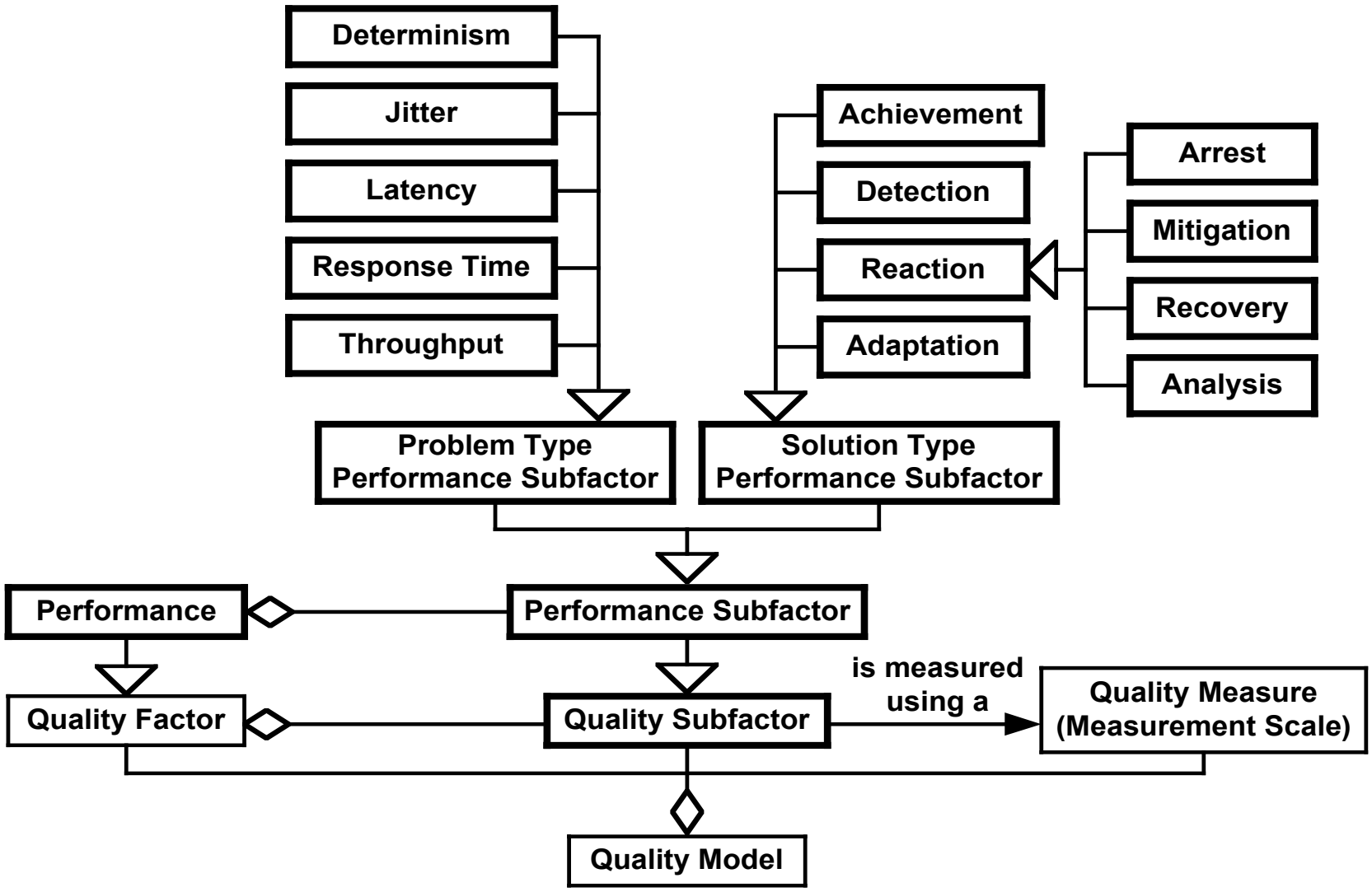
Quality Model



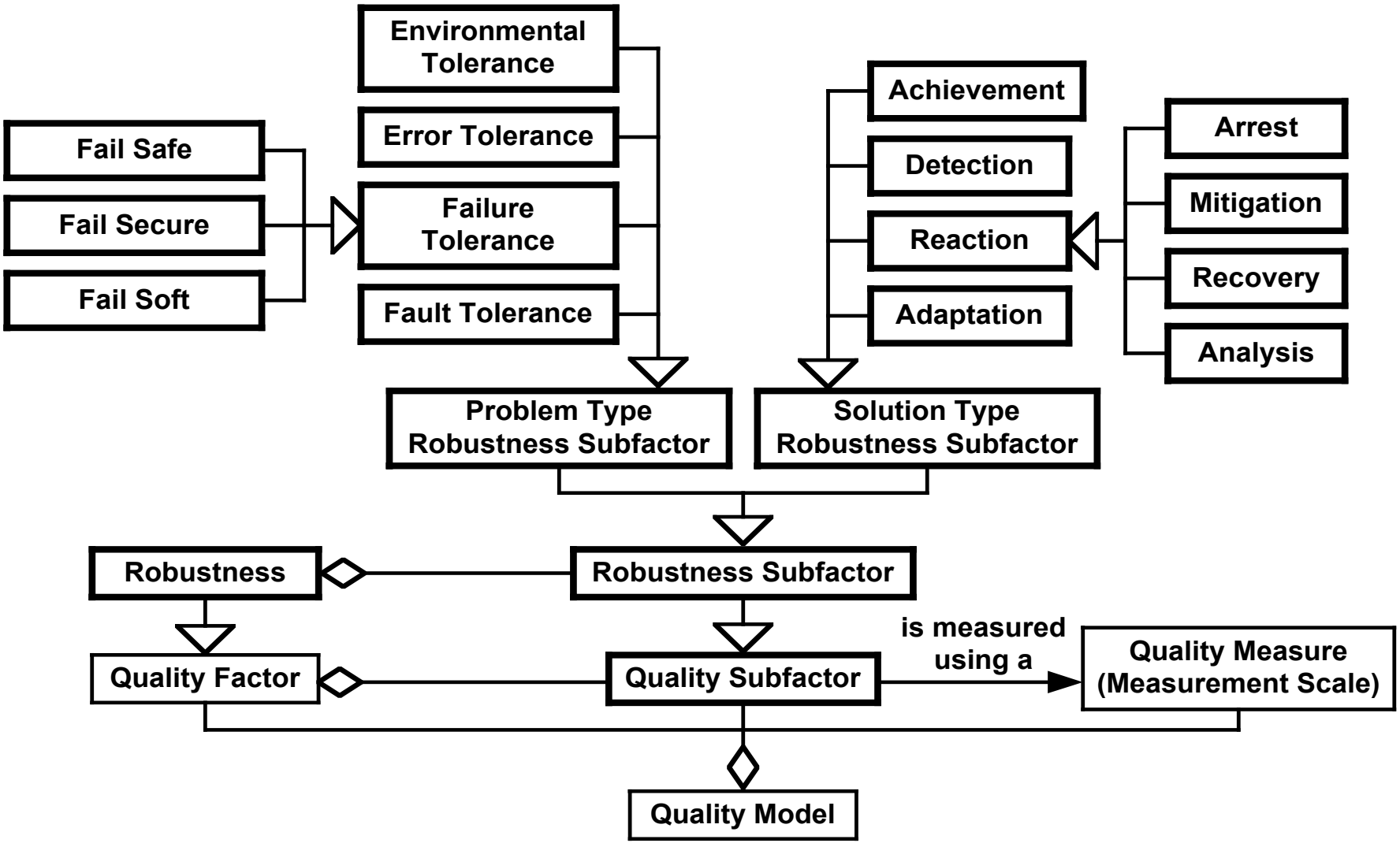
Quality Factors



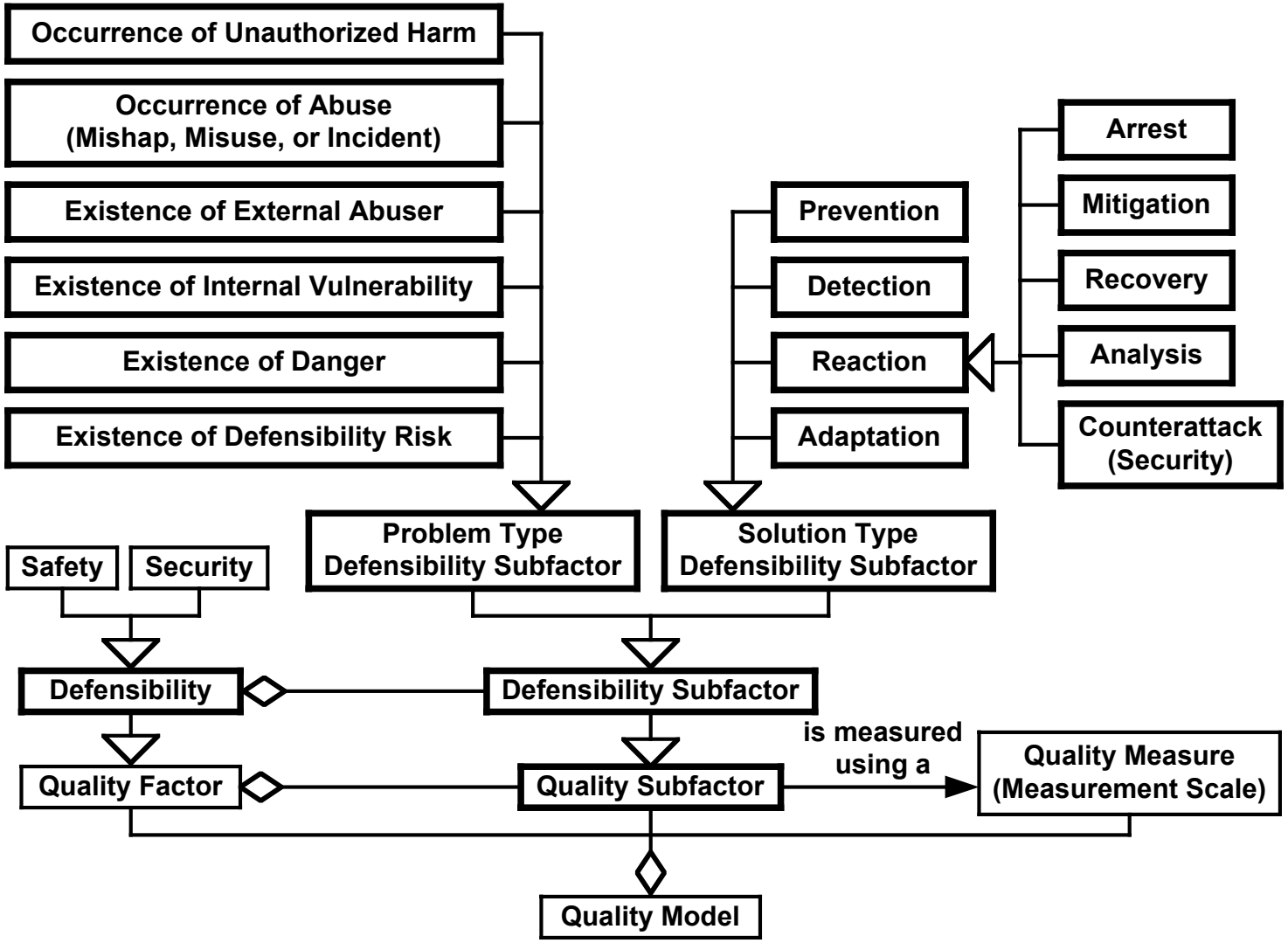
Performance Subfactors



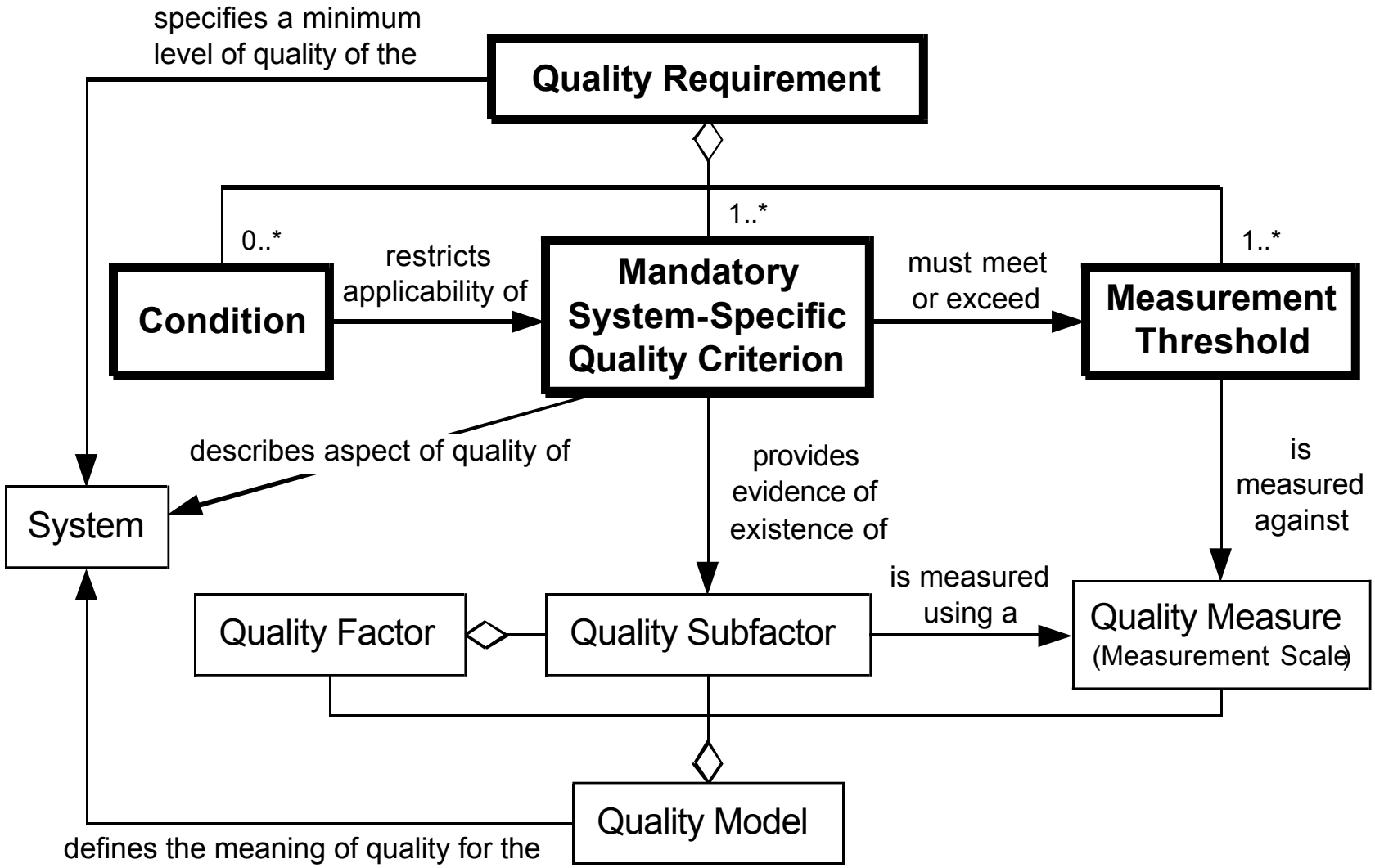
Robustness Subfactors



Defensibility Quality Subfactors



Components of a Quality Requirement



Example Quality Requirement

Hazard Prevention Safety Requirement:
"Under normal operating conditions, the subway shall not move when it's doors are open more than an average of once every 10,000 trips."

Component Parts:

- **Condition:**
  "Under normal operating conditions"
  (e.g., neither during maintenance nor a fire in a subway station)
- **Mandatory System-Specific Quality Criterion:**
  "the subway shall not *move* when it's doors are *open*"
  (The meaning of moving and open must be unambiguously defined.)
- **Measurement Threshold:**
  "more than an average of once every 10,000 *trips*."
  (A trip is defined as intentional travel from one subway station to the next.)

Importance of Measurement Threshold

Measurement Threshold is:

- Critical
- Difficult (but not impossible) to Determine
- Often left out of Quality Requirements
- Needed to Avoid Ambiguity

States *How Much* Quality is Necessary (adequate)

Enables Architect to:

- Determine if Architecture is Adequate
- Make Engineering Tradeoffs between Competing Quality Factors

Enables Tester to determine Test Completion Criteria

Safety and Security Engineering:

An Overview

You Are Here

Three Disciplines

Challenges

Requirements Engineering Overview

Safety and Security Engineering Overview _

Types of Safety- and Security-related Requirements

Common Consistent Collaborative Method

Conclusion

Similar Definitions

Safety Engineering

the engineering discipline within systems engineering concerned with lowering the risk of *unintentional unauthorized* harm to valuable assets to a level that is acceptable to the system's stakeholders by preventing, detecting, and reacting to such harm, mishaps (i.e., accidents and incidents), hazards, and safety risks

Security Engineering

the engineering discipline within systems engineering concerned with lowering the risk of *intentional unauthorized* harm to valuable assets to a level that is acceptable to the system's stakeholders by preventing, detecting, and reacting to such harm, misuses (i.e., attacks and incidents), threats, and security risks

Fundamental Safety and Security Concepts

Safety and Security as Quality Factors with associated Quality Subfactors

Valuable Assets

Unauthorized Harm to Valuable Assets

Stakeholders

Abuses (Accidents, Attacks, and Incidents)

Abusers (External and Internal, Malicious and Non-malicious)

Vulnerabilities (system-internal sources of dangers)

Dangers (Hazards and Threats)

Defensibility Risks (Safety and Security)

Goals, Policies, and Requirements

Defenses (Safeguards and Counter Measures)

Safety as a Quality Factor

Safety is the Subclass of Defensibility capturing the *Degree* to which:

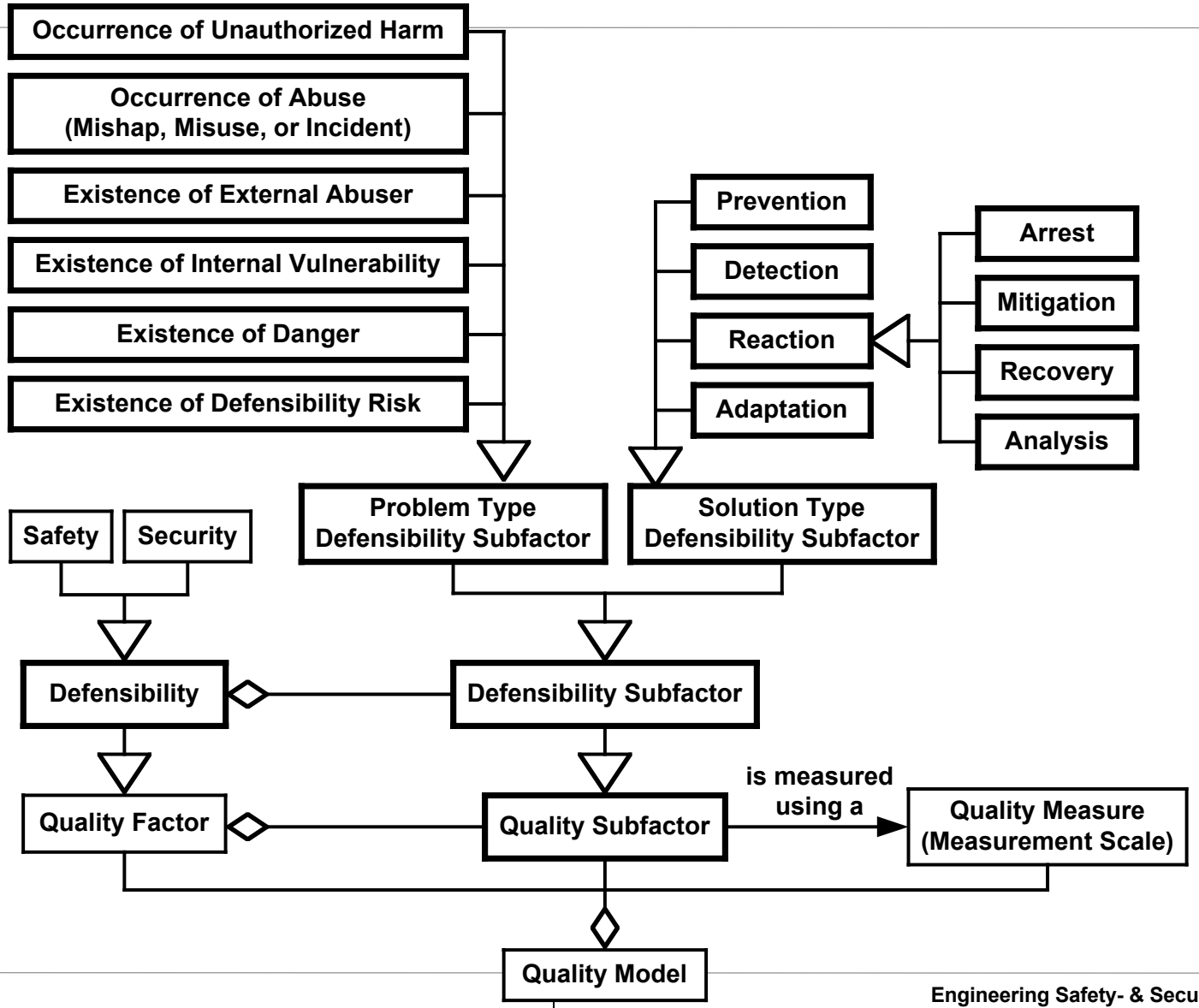
- *The Following (Safety Problems):*
  - *Accidental Harm* to Valuable Assets
  - *Safety Abuses (Mishaps) such as Accidents and Safety Incidents*
  - *Safety Abusers (People, Systems, and the Environment)*
  - *Safety Vulnerabilities*
  - *Safety Dangers (Hazards*) including the existence (conditions) of Nonmalicious Abusers who unintentionally exploit System Vulnerabilities to accidentally harm Vulnerable Valuable Assets
  - *Safety Risks*
- Are (Safety Solutions):
  - *Prevented (eliminated, mitigated, keep acceptably low)*
  - *Detected*
  - *Reacted to*
  - *Adapted to*

Security as a Quality Factor

Security is the Subclass of Defensibility capturing the *Degree* to which:

- *The Following (Security Problems):*
  - *Malicious Harm* to Valuable Assets
  - *Security Abuses (Misuses) such as Attacks and Security Incidents*
  - *Security Abusers (Attackers and Malware – systems, software, and hardware)*
  - *Security Vulnerabilities*
  - *Security Dangers (Threats)* including the existence (conditions) of Malicious Abusers who can exploit System Vulnerabilities to harm Vulnerable Valuable Assets
  - *Security Risks*
- Are (Security Solutions):
  - *Prevented (eliminated, mitigated, keep acceptably low)*
  - *Detected*
  - *Reacted to*
  - *Adapted to*

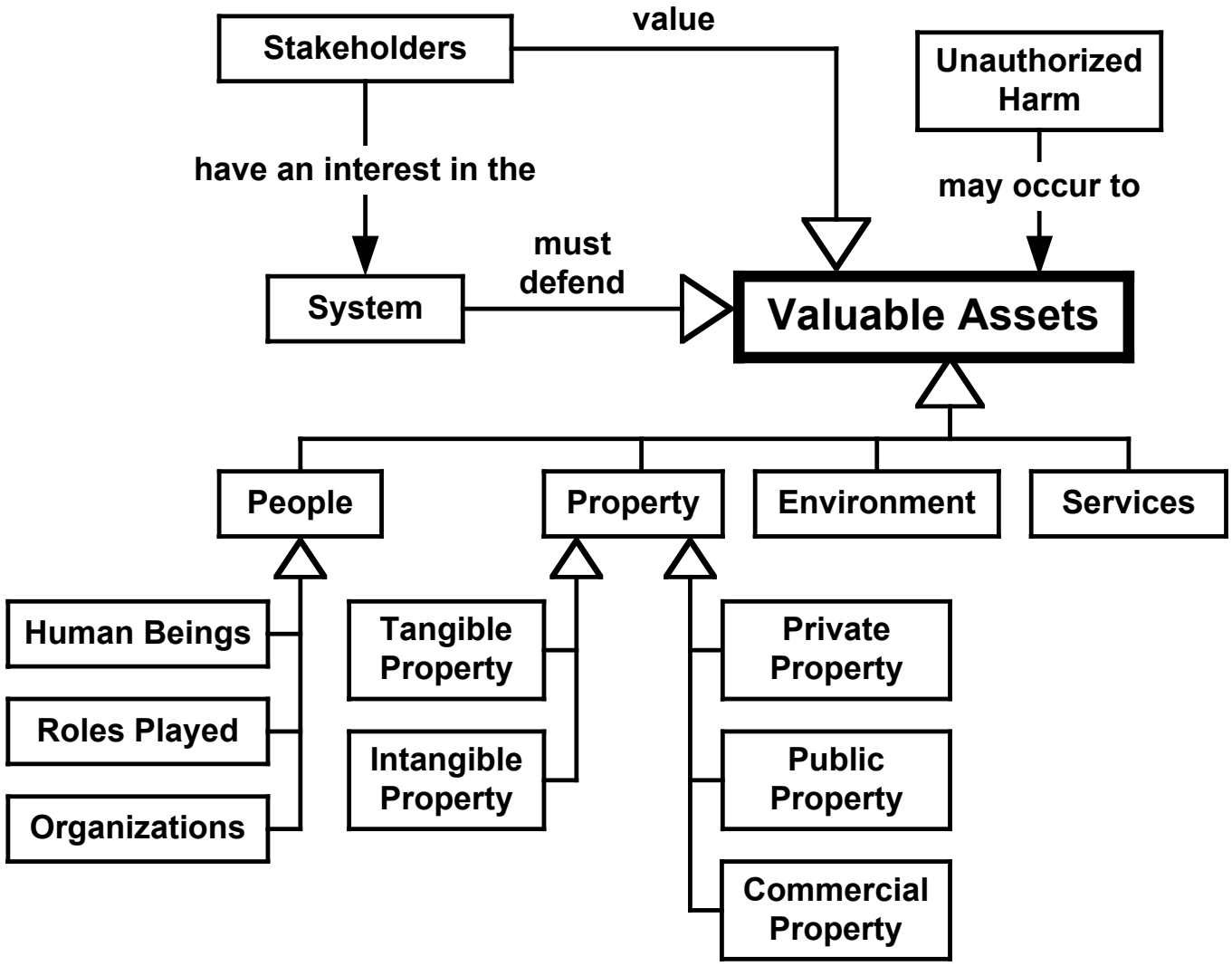
Defensibility Quality Subfactors



Different Types of Defensibility Requirements

	Unauthorized Harm	Abuse	Abuser	Vulnerability	Danger	Defensibility Risk
Prevention (current)	Prevent Occurrence of Unauthorized Harm	Prevent Occurrence of Abuse	Prevent Abuser Means or Opportunity	Prevent Existence of Vulnerability	Prevent Existence of Danger	Prevent Existence of Defensibility Risk
Detection (current)	Detect Occurrence of Unauthorized Harm	Detect Occurrence of Abuse	Detect Existence of Abuser	Detect Existence of Vulnerability	Detect Existence of Danger	Detect Existence of Defensibility Risk
Reaction (current)	React to Occurrence of Unauthorized Harm	React to Occurrence of Abuse	React to Existence of Abuser	React to Existence of Vulnerability	React to Existence of Danger	React to Existence of Defensibility Risk
Adaptation (future)	Adapt due to Unauthorized Harm	Adapt to Future Occurrence of Abuse	Adapt to Future Existence of Abusers	Adapt to Future Existence of Vulnerability	Adapt to Future Existence of Danger	Adapt due to Existence of Defensibility Risk

Valuable Assets



Categories of Asset Values

Assets Valuable to any System Stakeholder, not just Organization or System Owner [ISO/IEC 13335-1].

The Values of Assets are used to determine how to invest Limited Resources in protecting Assets from Unauthorized Harm.

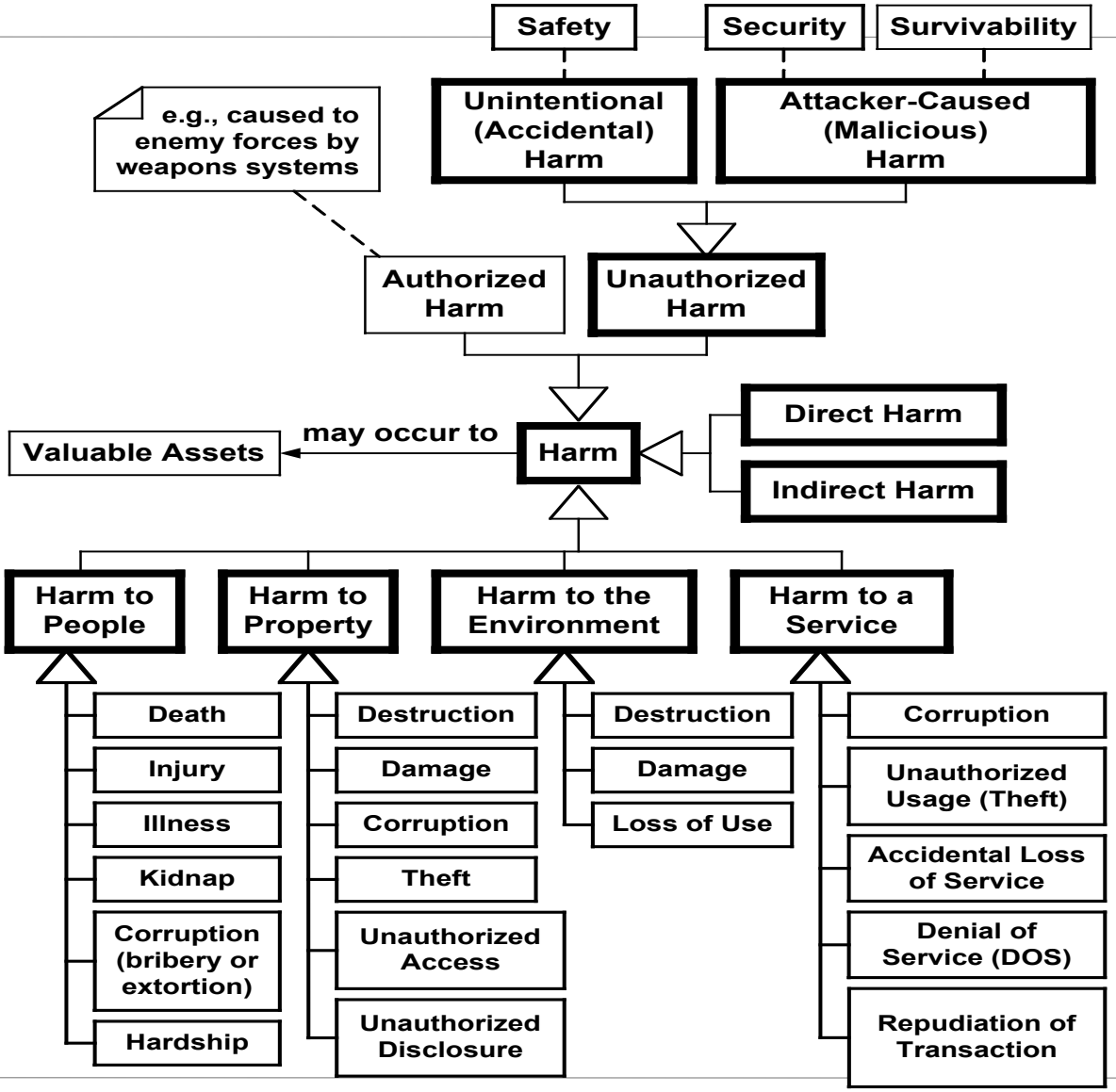
Sometimes, All Values are measured in terms of Money to enable comparison of “Apples and Oranges.”

More often, the Asset Values are Categorized:

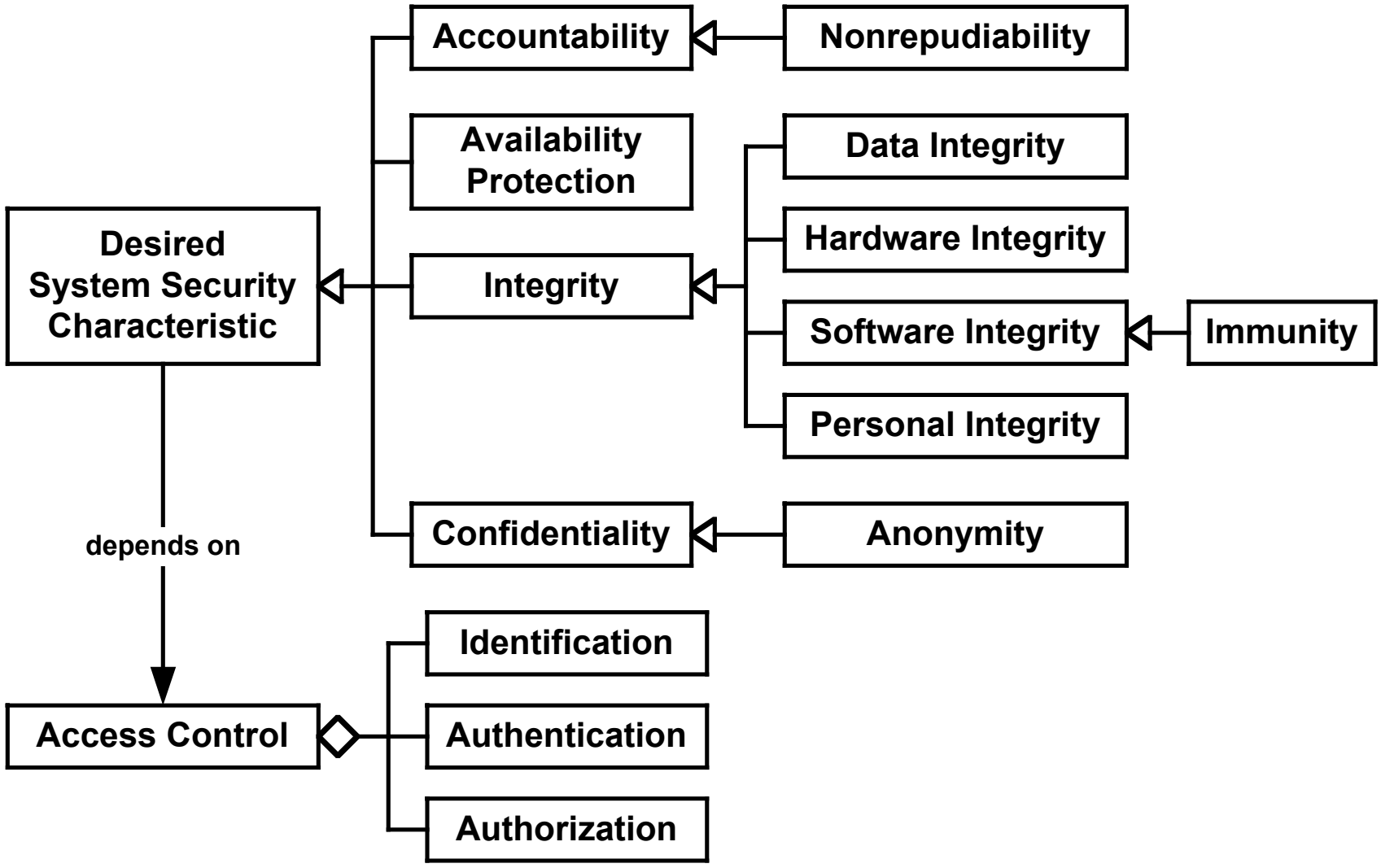
- Extremely valuable (i.e., invaluable or priceless)
- Major
- Moderate
- Minor
- Negligible (i.e., not worth considering)

When used, Categories should be Well Defined (e.g., Unambiguous)

Types of Harm



Security Characteristics as Types of Harm



Harm Severity

Harm severity is an appropriate categorization of the amount of harm.

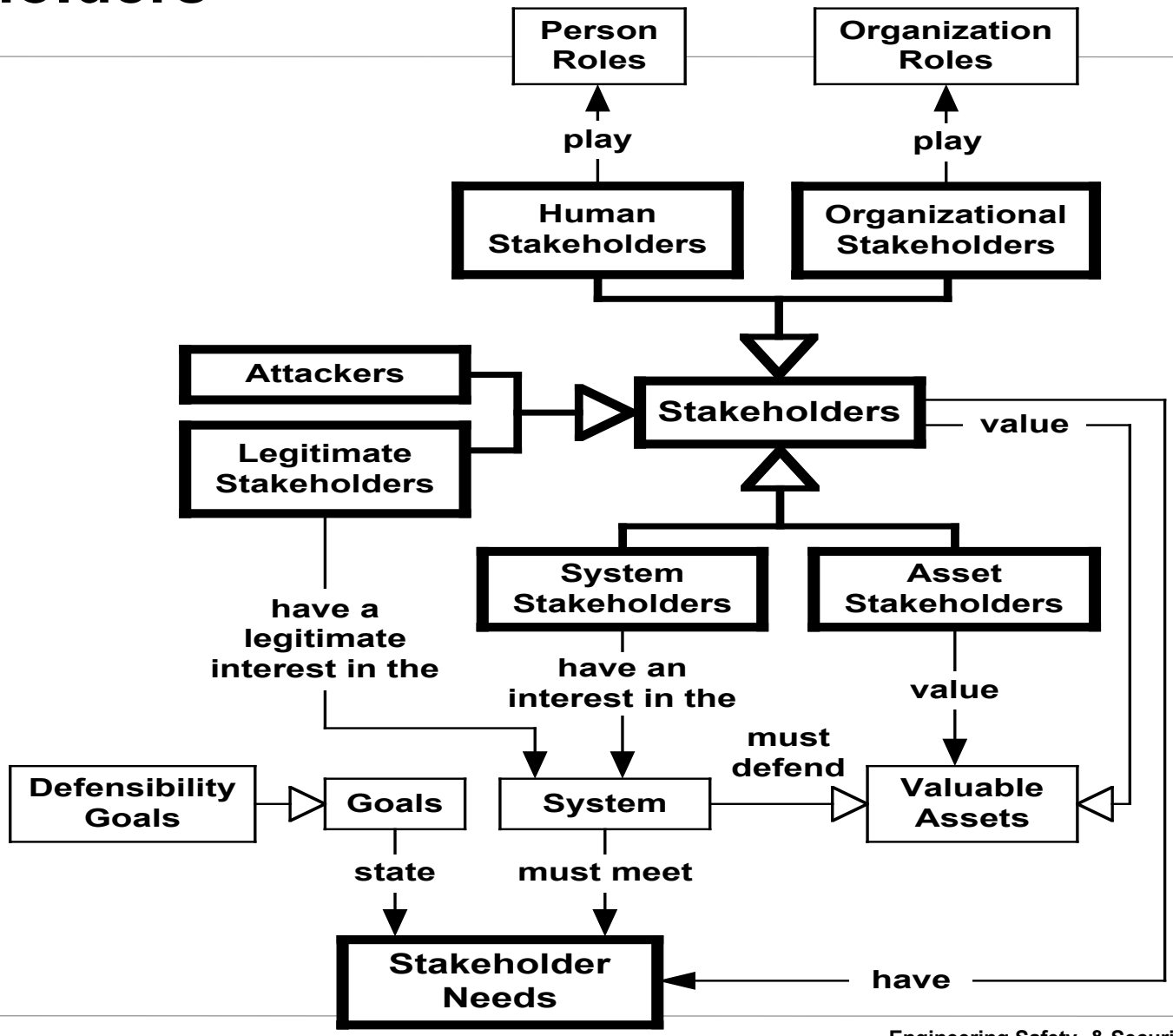
Harm severity categories can be standardized (ISO, military, industry-wide) or endeavor-specific.

Harm severity categories need to be:

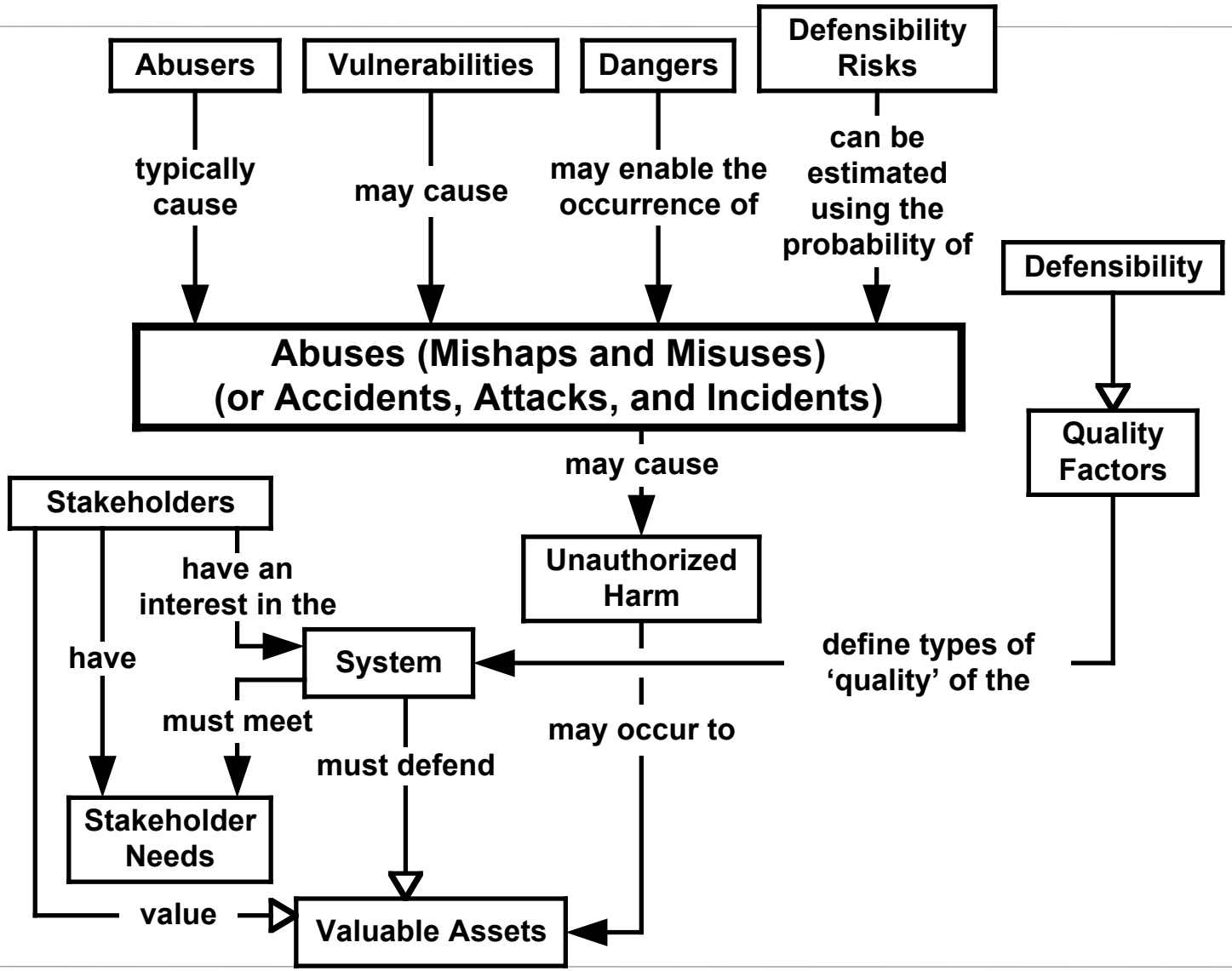
- Clearly identified.
- Appropriately and unambiguously defined.

Note that some standards confuse harm severity with hazard "severity" (i.e., categorization of hazard based on the severity of harm that its accidents can cause)

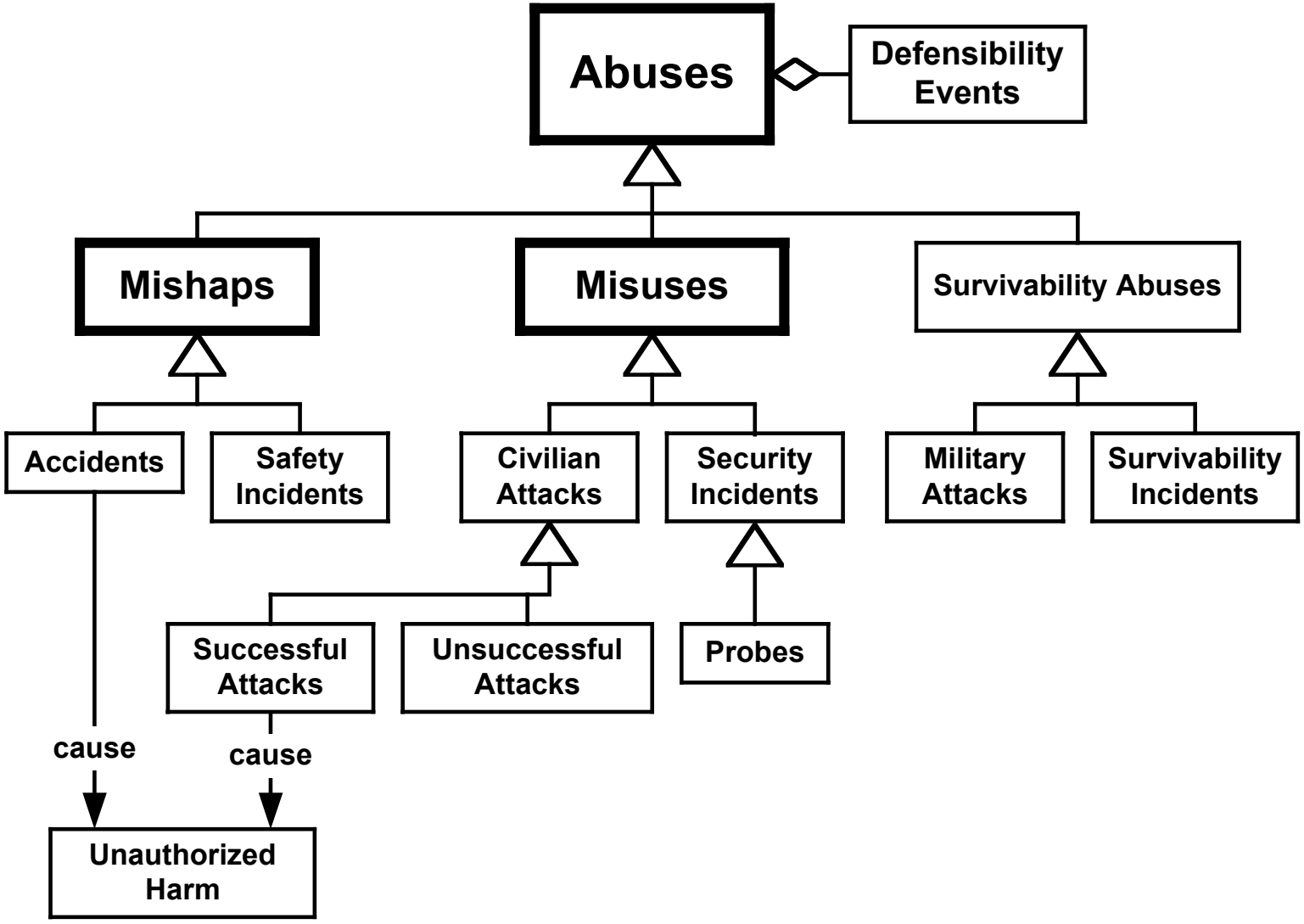
Stakeholders



Abuses



Types of Abuses



Importance of Accidents

Accidents can have expensive and potentially fatal repercussions:

- Ariane 5 Maiden Launch
  - Reuse of Ariane 4 software not matching Ariane 5 specification
- Mars Climate Orbiter ($125 million)
  - English vs. Metric units mismatch
- Mars Polar Lander
  - Missing requirement concerning touchdown sensor behavior
- Therac–25 Radiation Therapy Machine
  - Timing of unusual input sequence results in unexpected output
- Patriot Missile Battery Misses SCUD
  - Missing availability (uptime) requirement

Abuse Likelihood Categories

Abuse Likelihood Categorization is an appropriate categorization of the probability that an abuse occurs.

Abuse Likelihood Categories:

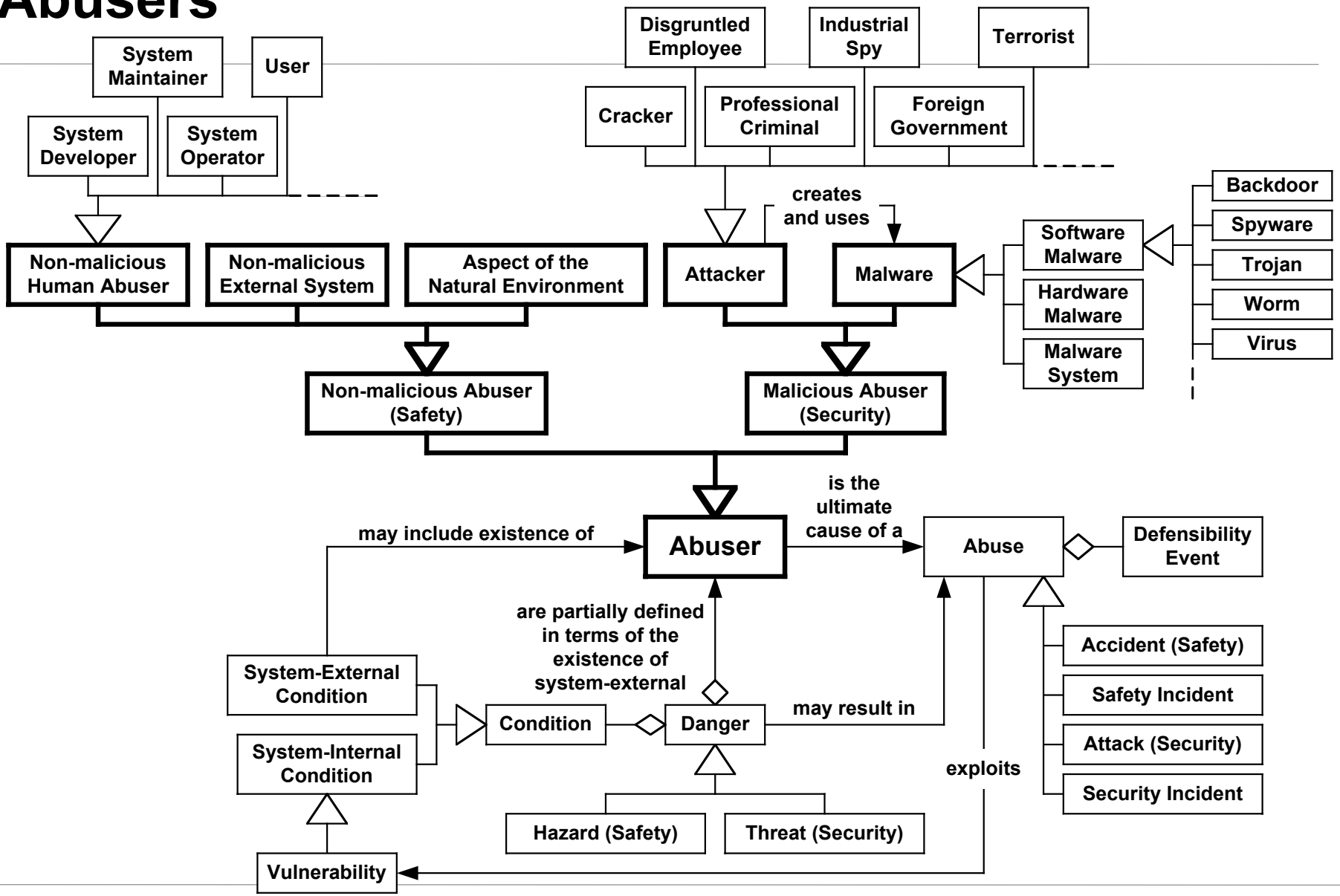
- Can be standardized (ISO, military, industry-wide) or endeavor-specific.
- Need to be identified and defined.

Example Abuse Likelihood Categories might include:

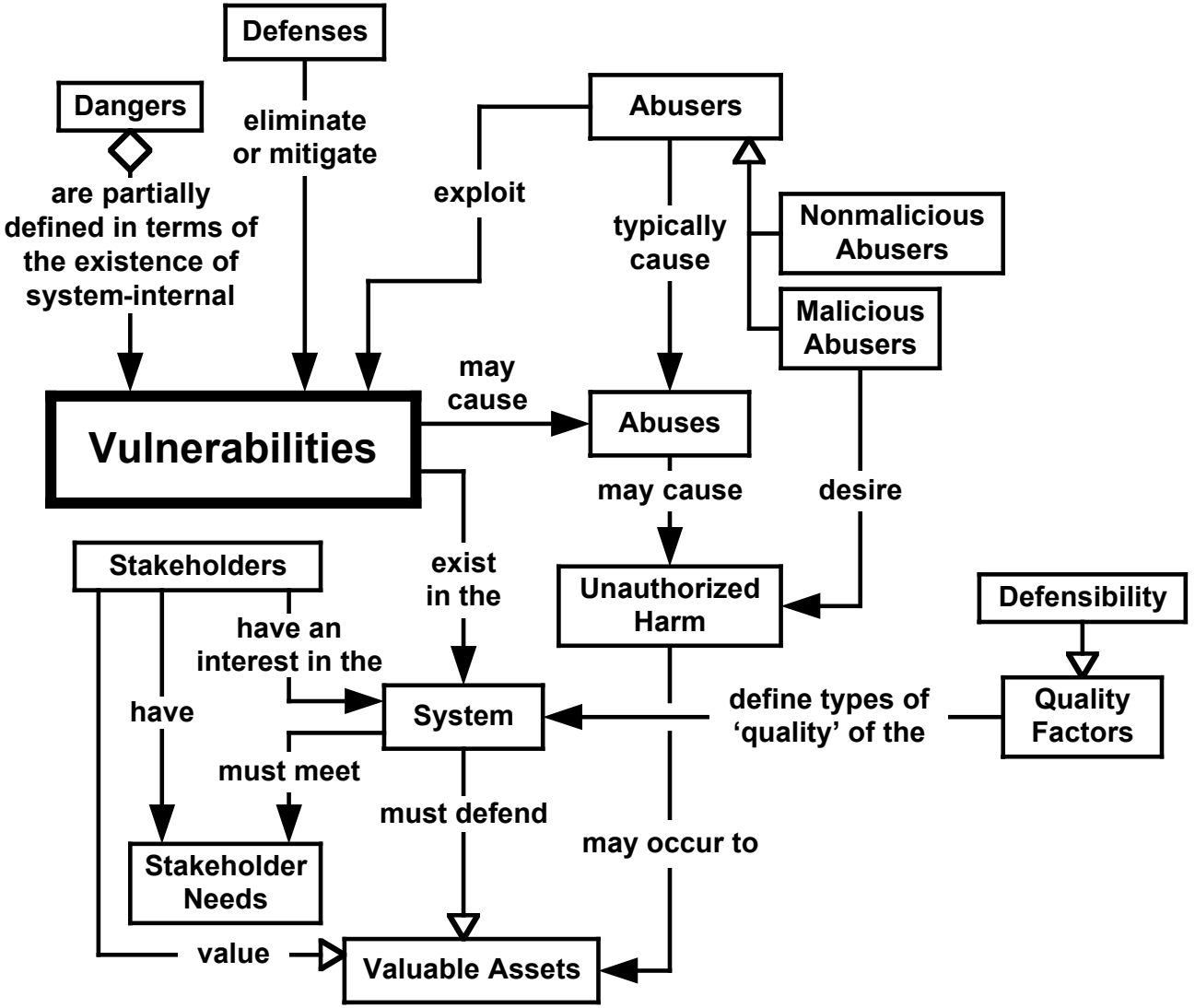
- Frequent
- Probable
- Occasional
- Remote
- Implausible

Abuse Likelihood Categories need to be carefully and unambiguously defined.

Abusers



Vulnerabilities



Vulnerabilities

Vulnerability

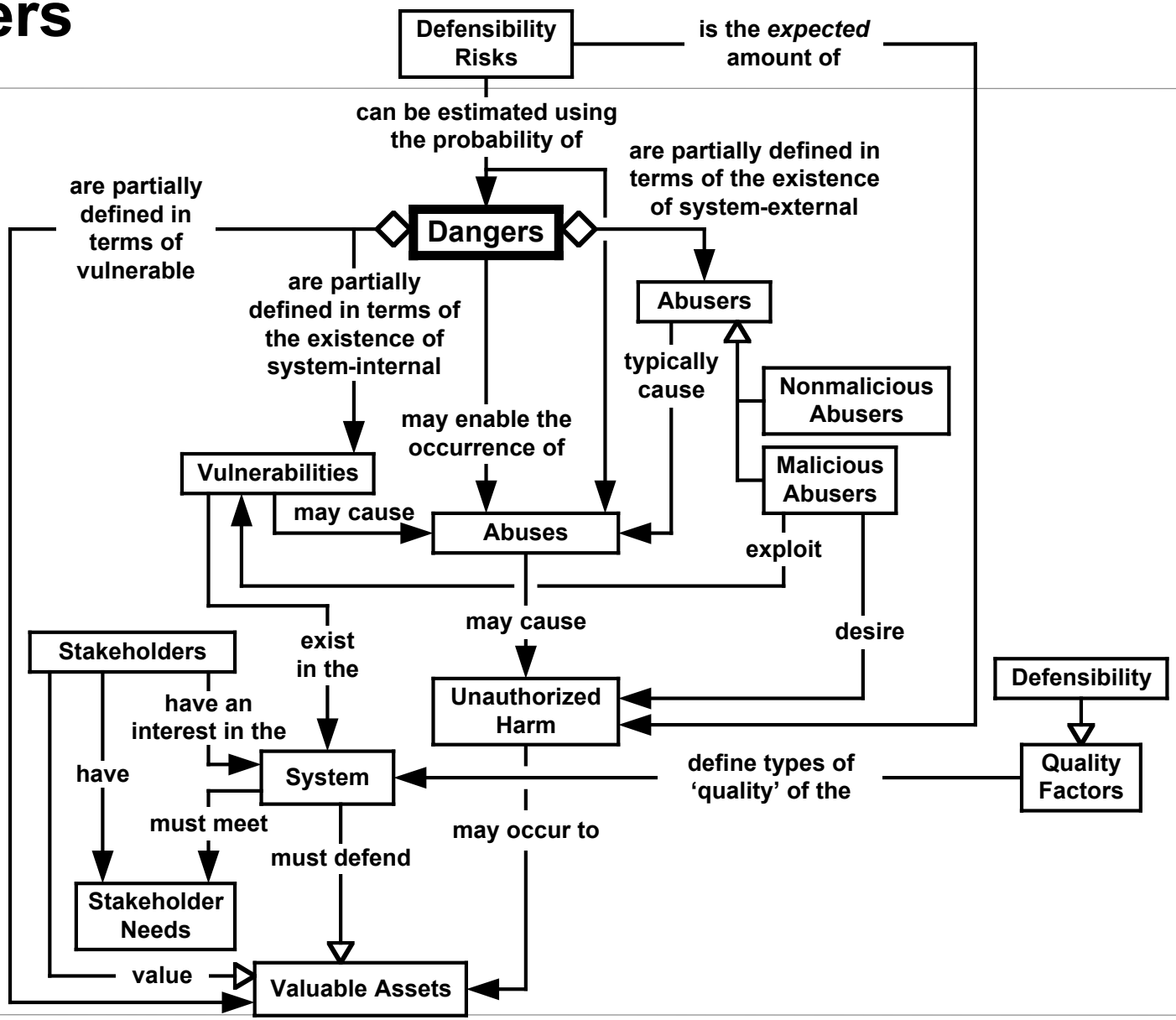
a potential or actual *system-internal weakness* (defect) in the architecture, design, implementation, integration, or deployment of a system that enables:

- A Danger (Hazard or Threat) to Exist.
- An Abuse (Mishap or Misuse) to Occur.

Ways to Identify Vulnerabilities include:

- Analyze Historical Data
- Identify Hardware or Software Defects
- Consider Hardware or Software Failures that can cause Vulnerabilities.

Dangers



Dangers

Danger

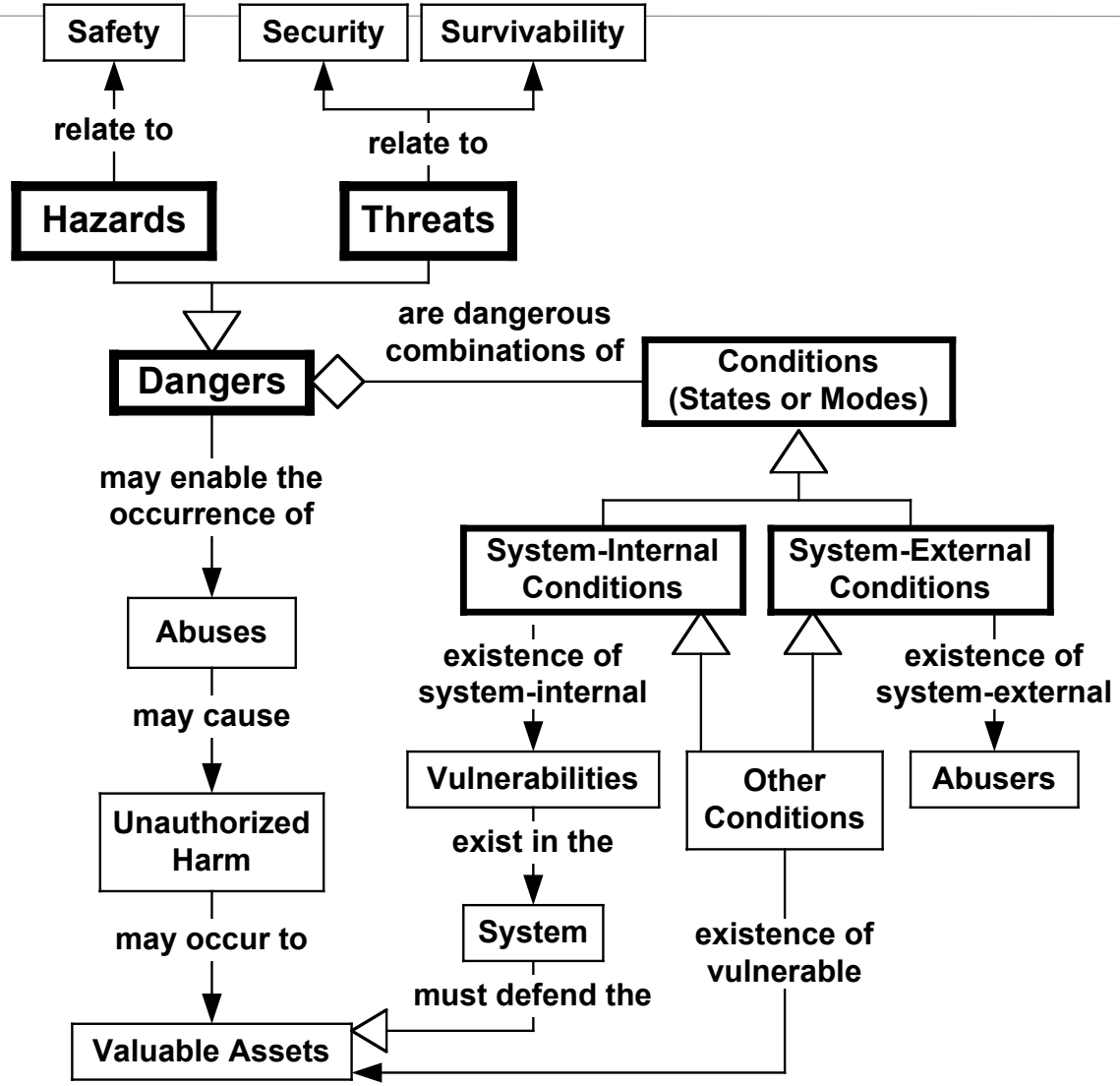
a potential or actual situation that can cause or contribute to the occurrence of one or more related abuses

A danger consists of the *existence* of one or more of the following *conditions*:

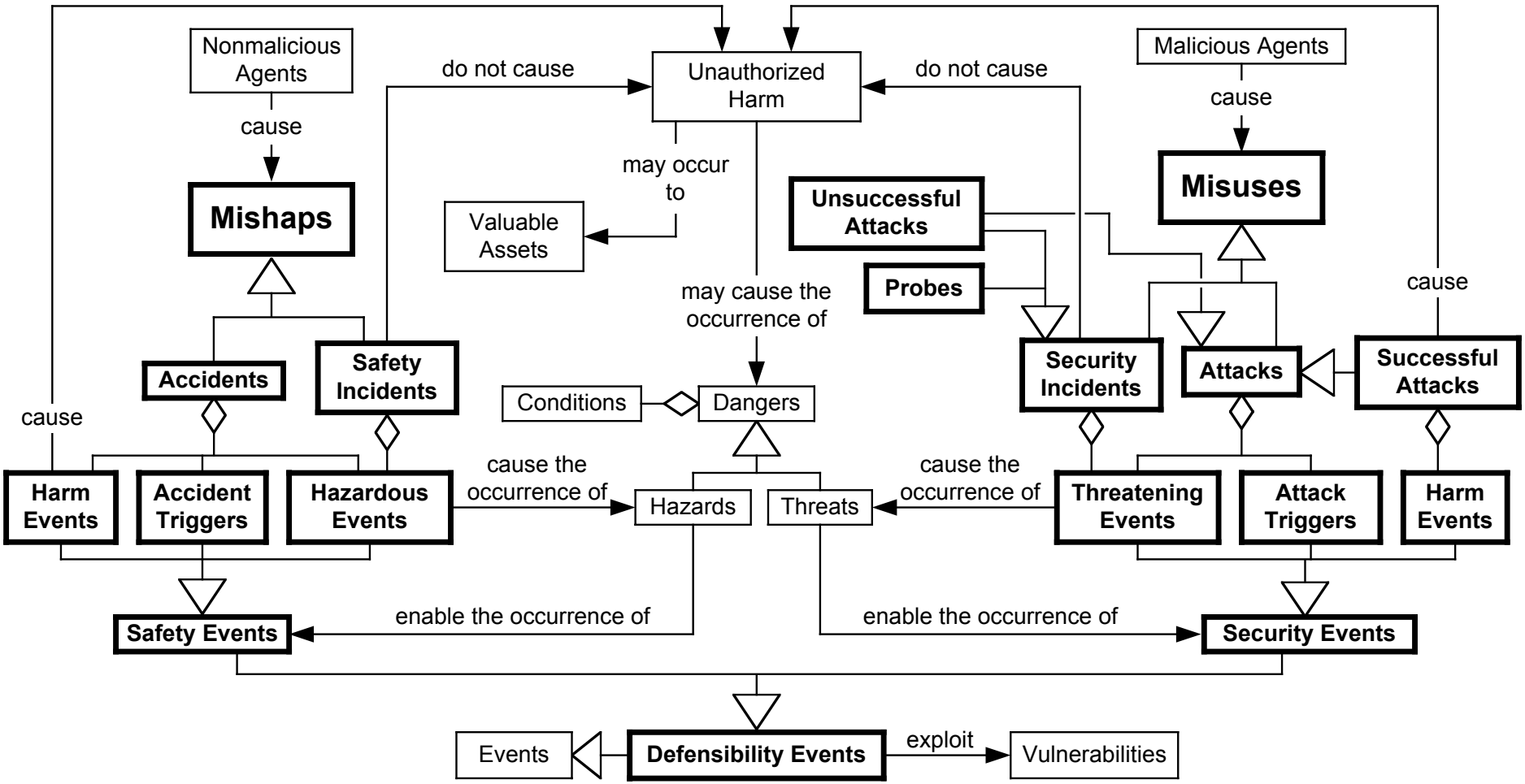
- Vulnerable valuable assets that can be harmed by the abuses
- System-*internal* vulnerabilities or other system-*internal* conditions, states, or modes
- System-*external* abusers or other system-external conditions, states, or modes

Dangers are sets of Conditions, not Events.

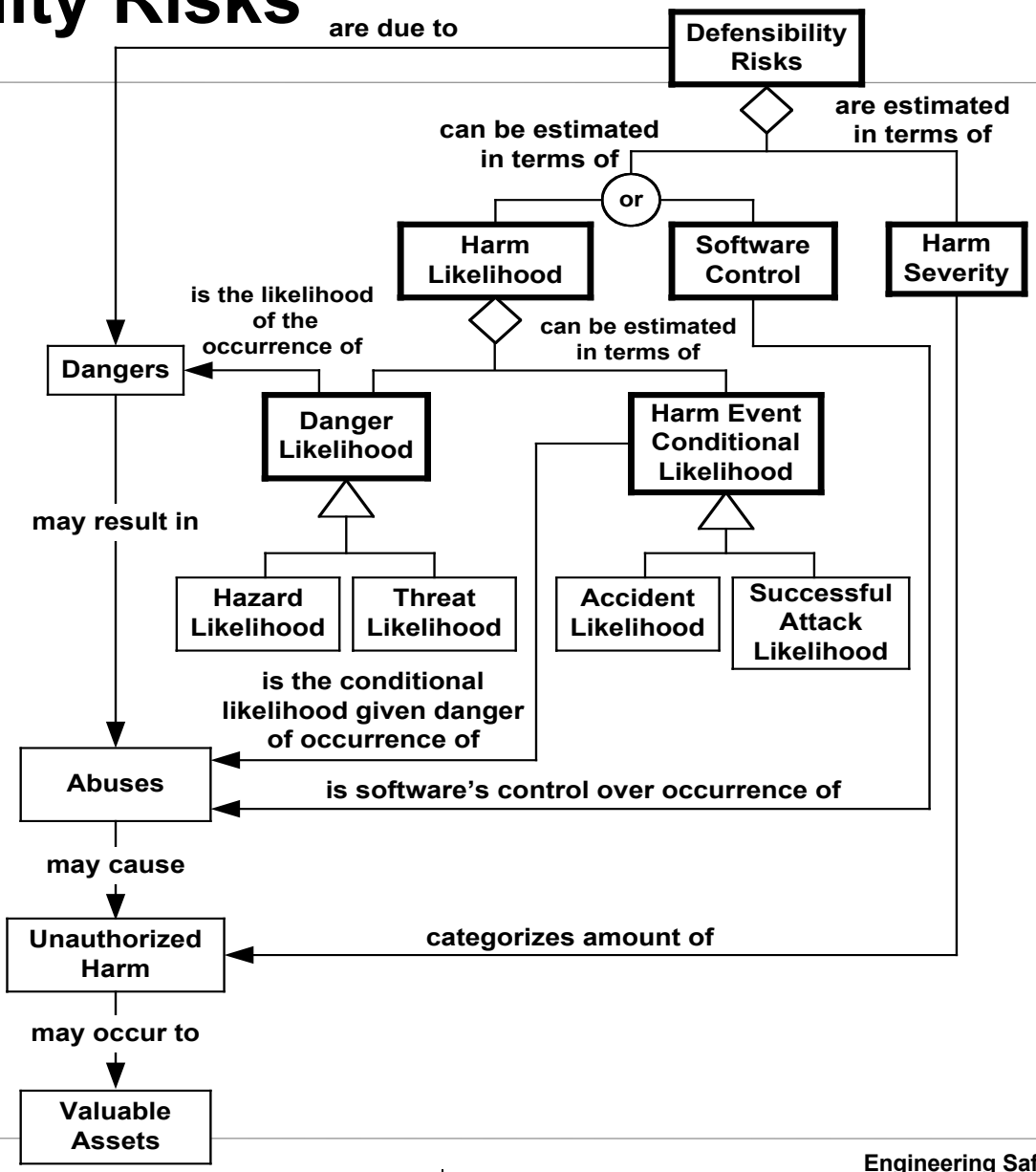
Types of Dangers



Mishaps and Misuses vs. Hazards and Threats



Defensibility Risks



Defensibility Risks

Risk

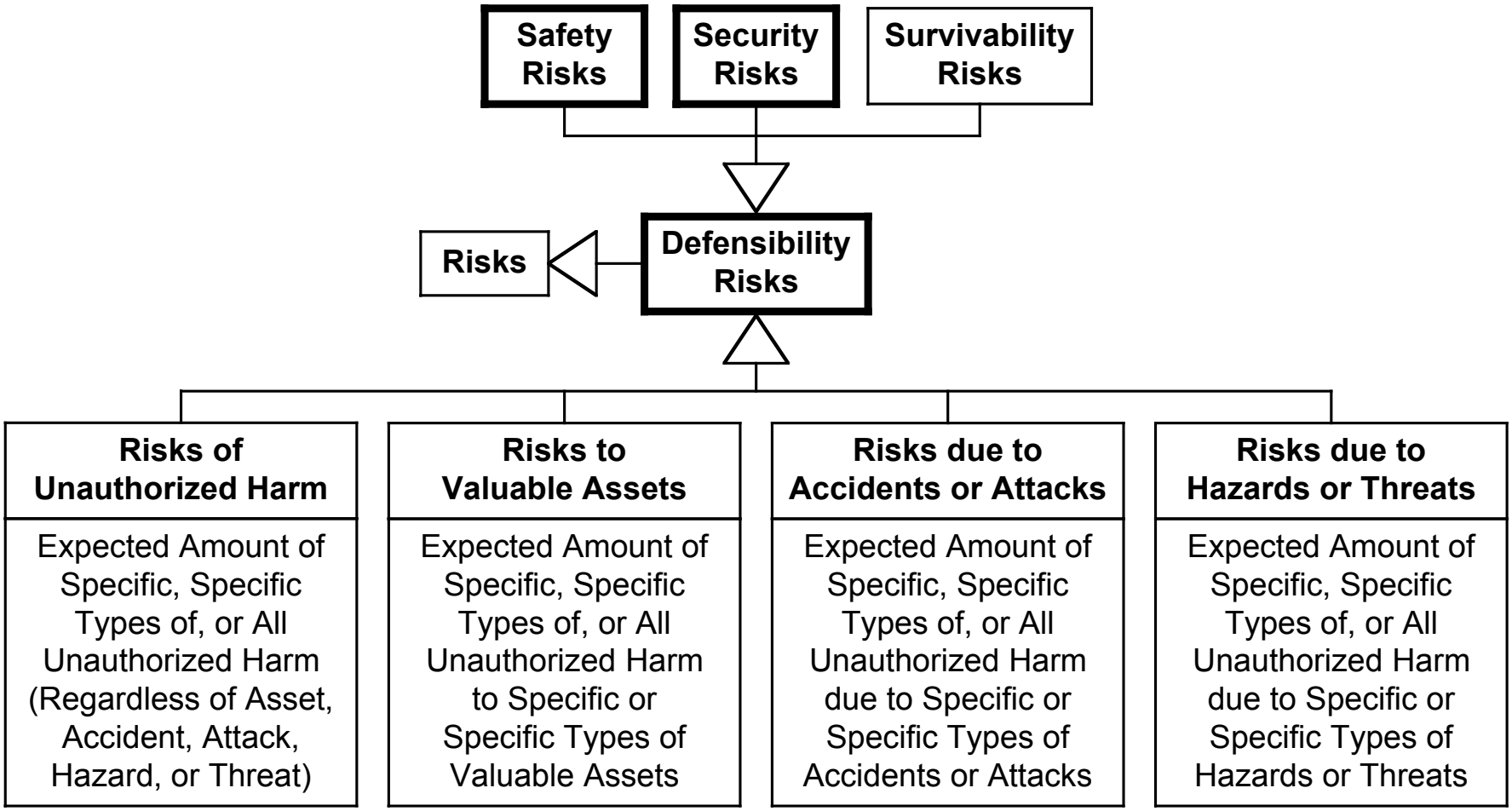
the [maximum credible] *Expected* Amount of Unauthorized Harm

Defensibility Risk

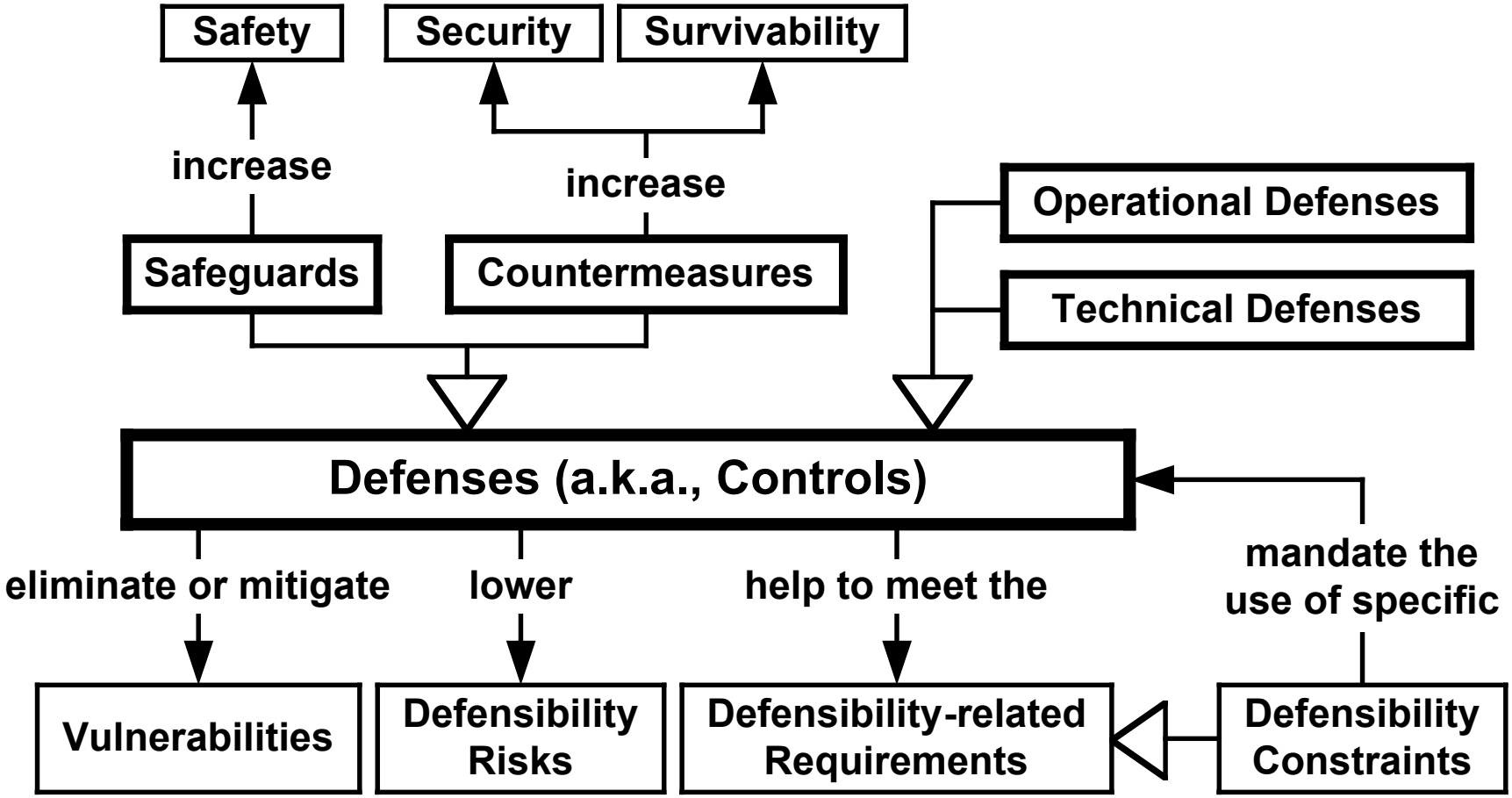
the expected amount of harm that can occur

- To [a type of] Valuable Assets
- Due to a specific [type] of Abuse
- Because of the existence of [a type of] Vulnerability
- Because of the existence of a type of Abuser
- Because of the existence of [a type of] Danger

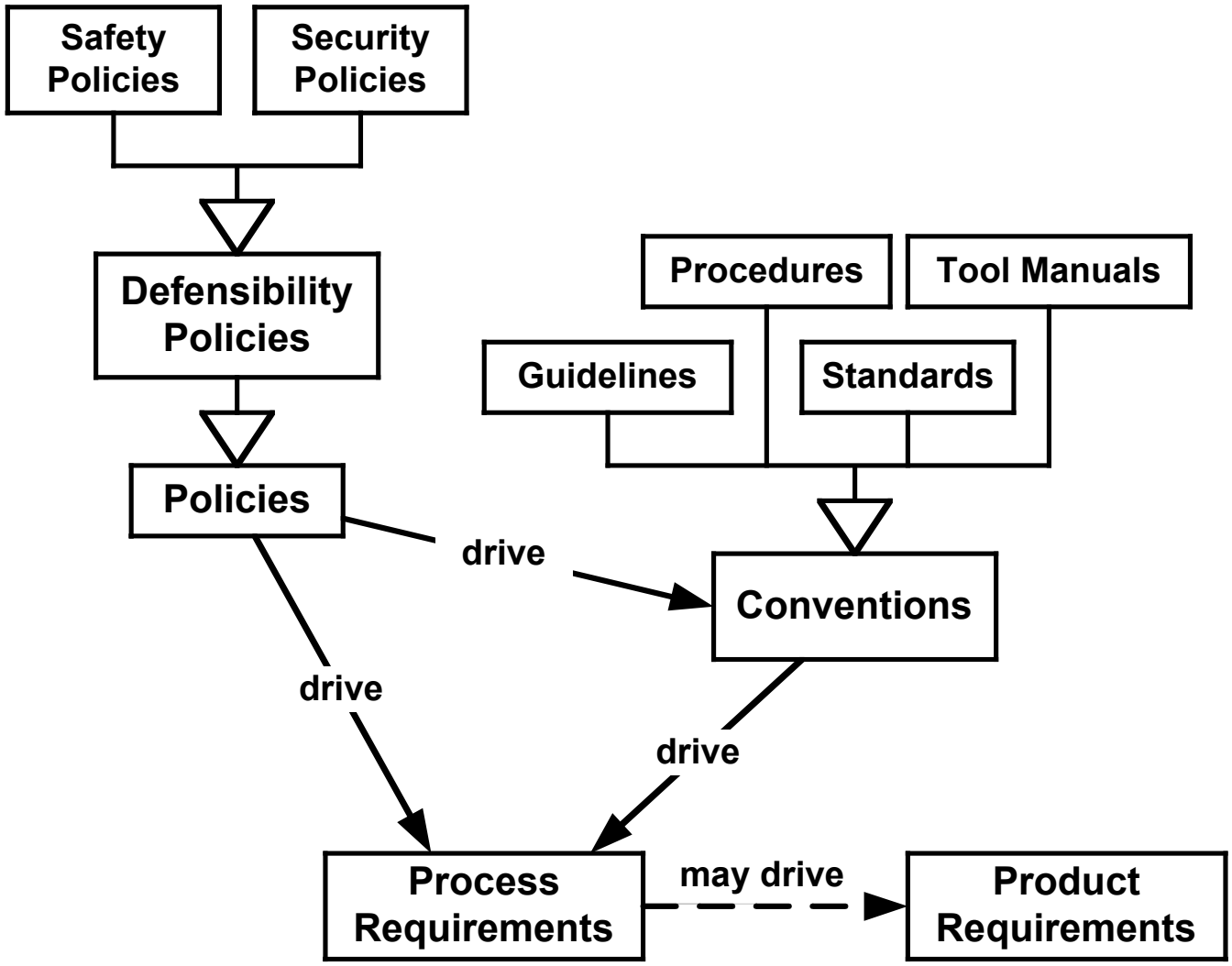
Types of Risks



Types of Defenses



Safety and Security Policies and Conventions



Safety and Security Policies

Policy

a Strategic *Process* Mandate that establishes a desired Goal

Defensibility Policy

a Policy that enables the Achievement of one or more *Safety* or *Security Goals*

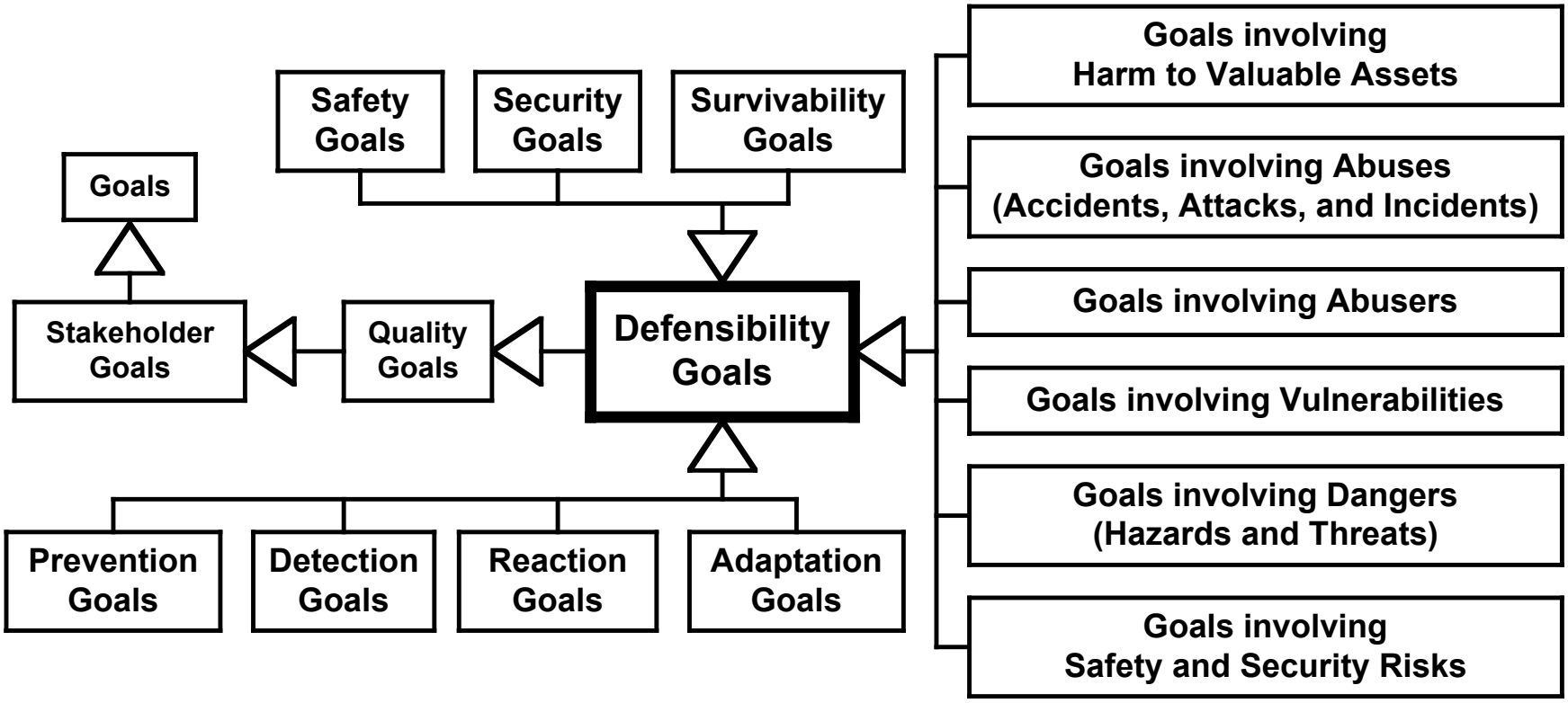
Examples

- "The overall responsibility for safety must be identified and communicated to all stakeholders."
- "A hazard analysis shall be performed during early in the project."
- "All users will have safety training."

Policies are typically collected into Safety or Security Policy Documents.

In practice, policies are confused with requirements, and conversely policy documents may sometimes include requirements.

Types of Defensibility Goals



Types of Safety- and Security-Related Requirements

You Are Here

Three Disciplines

Challenges

Requirements Engineering Overview

Safety and Security Engineering Overview

Types of Safety- and Security-related Requirements _

Common Consistent Collaborative Method

Conclusion

Types of Safety- and Security-Related Requirements

Too often only a Single Type of Requirements is considered.

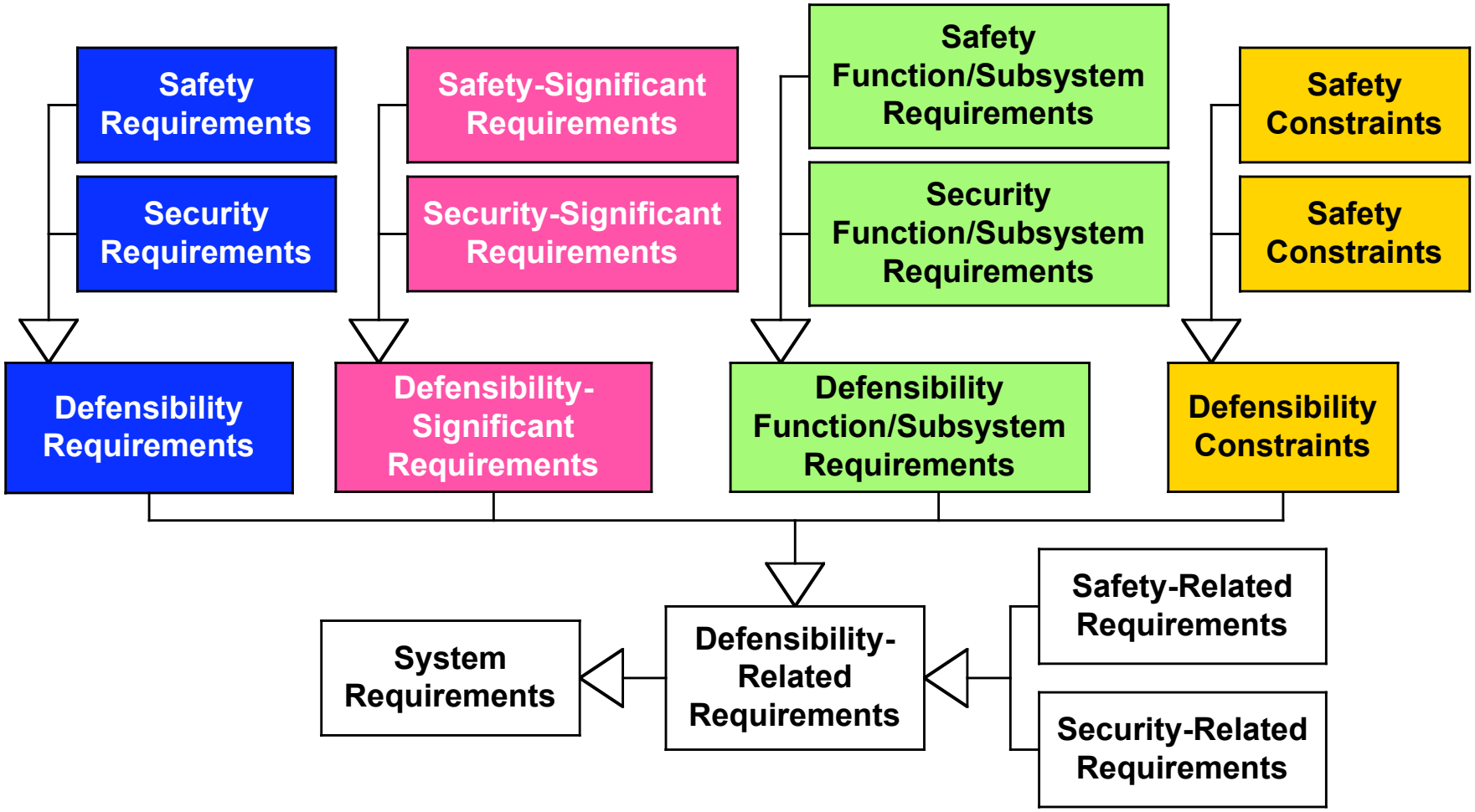
Not just:

- Special Non-Functional Requirements (NFRs):
  - Safety and Security Requirements are Quality Requirements
- Safety- and Security-*Significant* Functional, Data, and Interface Requirements
- Constraints:
  - on Functional Requirements
  - Architecture and Design Constraints
- Safety and Security Functions/Subsystems

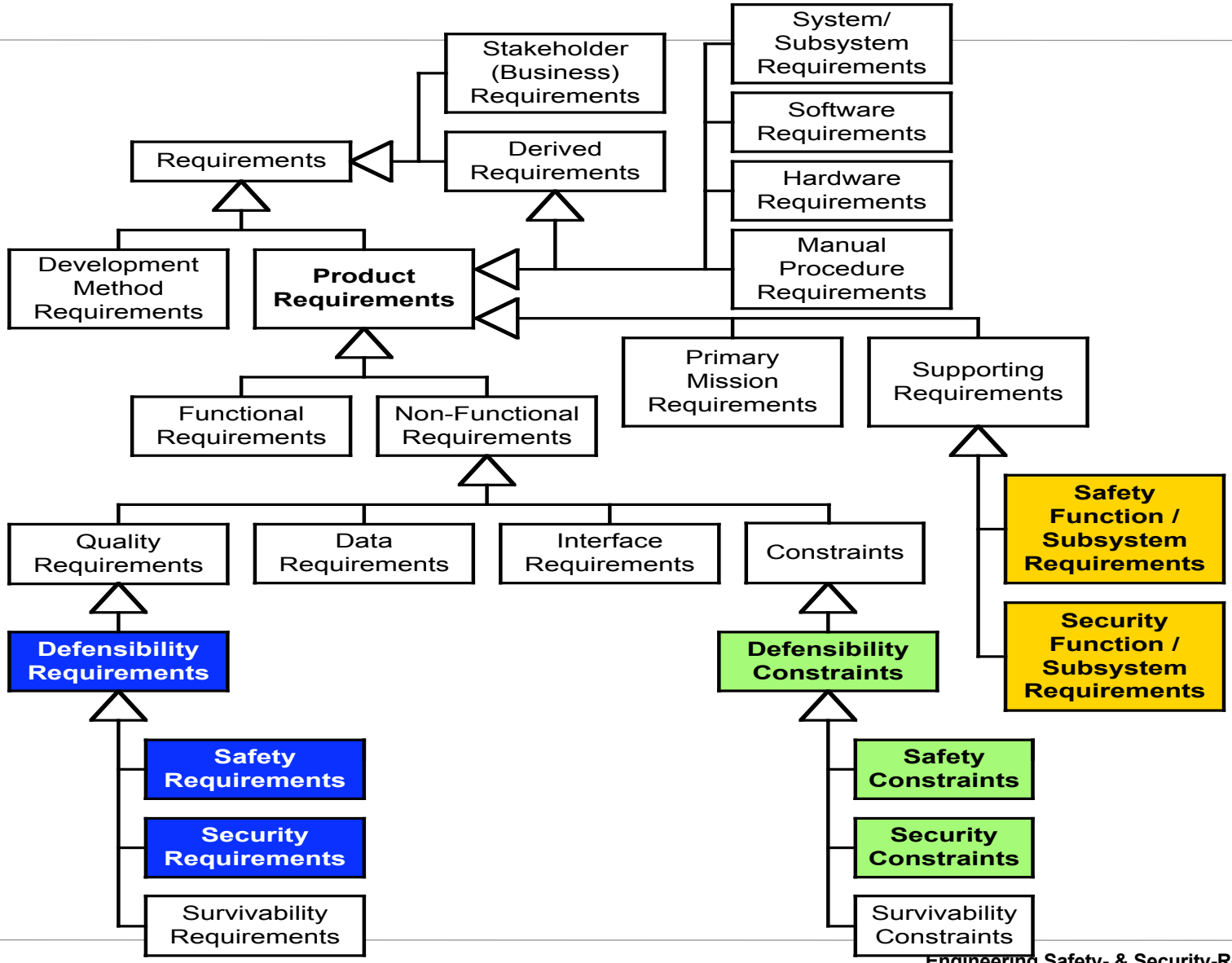
Reason for Presentation Title

Safety- and Security-Related Requirements for Software-Intensive Systems

Types of Defensibility-Related Requirements



Types of Safety- and Security-Related Requirements



Safety and Security Requirements

Safety and Security Requirements are *Quality* Requirements.

Quality Requirements are Product Requirements that specify a Mandatory Minimum Amount of a Type of Product Quality:

- Quality Factor
- Quality Subfactor

Quality Requirements should be:

- Scalar (How Well or How Much)
- Based on a Quality Model
- Specified in Requirements Specifications and NOT in:
  - Secondary Specifications
  - Safety/Security Documents

Quality Requirements are Critically Important Drivers of the Architecture.

Example *Safety* Requirement Templates

"When in mode V, the system shall not cause *accidental harm* of type W to valuable assets of type X at an average rate of more than Y asset value per Z time duration."

"When in mode W, the system shall not cause *mishaps* of type X with an average rate of more than Y mishaps per Z trips."

"When in mode X, the system shall not cause *hazard* Y to exist more than an average of Z percent of the time."

"When in mode X, the system shall not have a *safety risk level* of Y."

"When in mode X, the system shall *detect accidents* of type Y an average of at least Z percent of the time."

"Upon detecting an accident of type W when in mode X, the system shall *react* by performing functions Y an average of at least Z percent of the time."

Example *Security* Requirement Templates

"When in mode V, the system shall limit the occurrence of *malicious harm* of type W to valuable assets of type X to an average rate of less than Y asset value per Z time duration."

"When in mode W, the system shall prevent the first *successful attacks* of type X for a minimum of Z time duration."

"When in mode X, the system shall not have *security vulnerability* Y for more than an average of Z percent of the time."

"When in mode X, the system shall not have a *security risk level* of Y."

"When in mode X, the system shall *detect misuses* of type Y an average of at least Z percent of the time."

"Upon detecting a misuse of type W when in mode X, the system shall *react* by performing functions Y an average of at least Z percent of the time."

Safety- and Security-Significant Requirements

Defensibility-Significant Requirement

a requirement with significant safety or security ramifications

Are identified based on Safety or Security (e.g., Hazard or Threat) Analysis

Subset of non-Safety and non-Security Requirements:

- Functional Requirements
- Data Requirements
- Interface Requirements
- Other Quality Requirements
- Constraints

SALs and SEALs

Safety/Security Assurance Level (SAL)

- a Category of Requirements based on the degree of Safety or Security Relevance

Safety/Security Evidence Assurance Level (SEAL)

- a Category of Architectural Components based on the highest SAL of the Allocated and Derived Requirements they implement

SALs categorize *Requirements.*

SEALs categorize *Architectural Components* that helps fulfill these Requirements.

SEALs define increasingly strict associated Development Methods needed to assure implementation of the highest associated SAL.

SALs and SEALs

Safety- and Security-Significant Requirement has SAL > 0:

- May have minor Safety/Security Ramifications
- May be Safety- or Security-Critical
- May have intolerable Safety or Security Risk

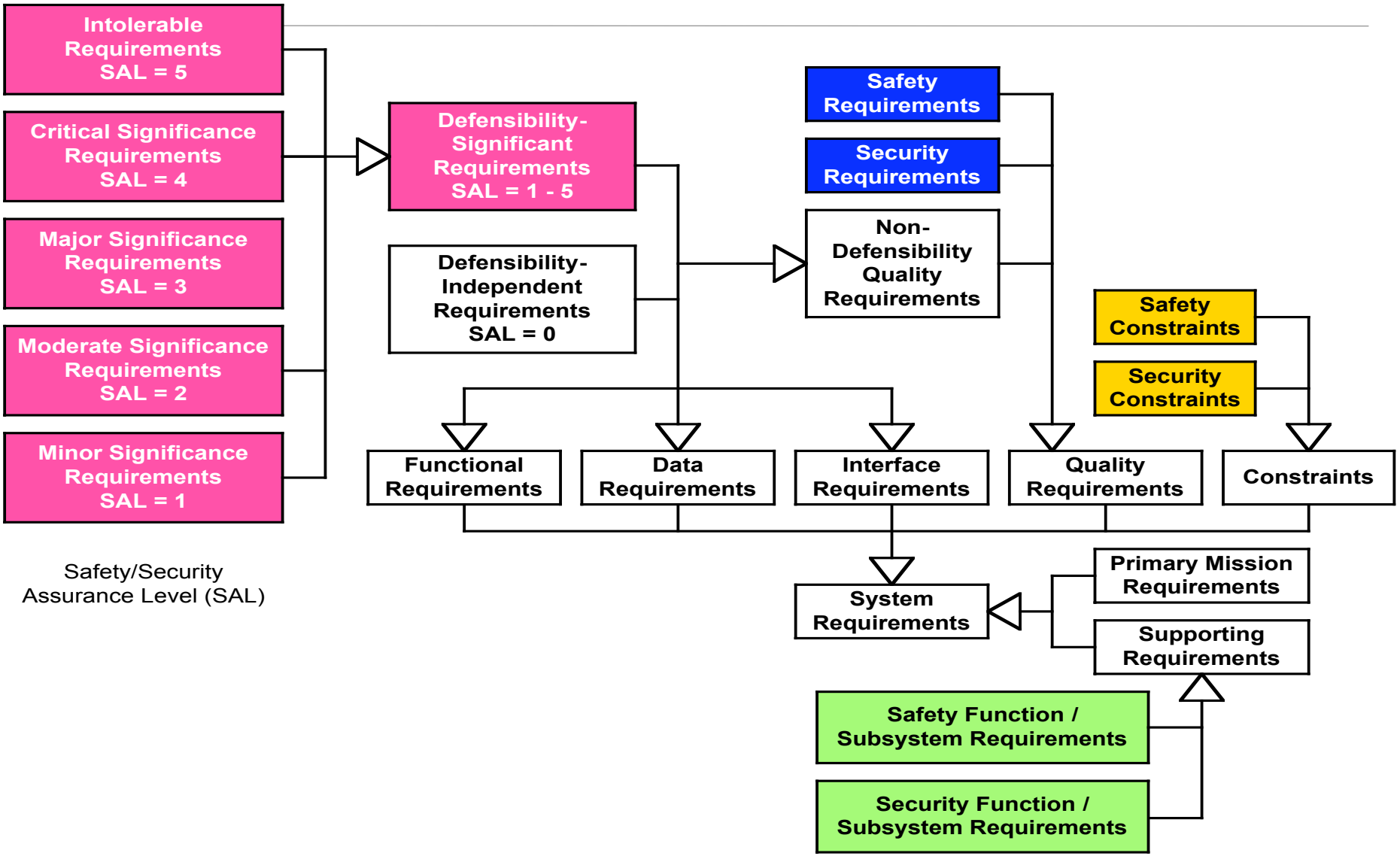
SAL is also known as:

- Safety Integrity Levels (SIL):
  - IEC 61508 and IEC 61511
  - UK Defence Standard 00-56 Issue 2

SEAL is also known as:

- Assurance Evidence Level (AEL) – CAP670 (Safety)
- Evaluation Assurance Level (EAL) - Common Criteria (Security)
- Evaluation Level (E) – Information Technology Security Evaluation Criteria (ITSEC)

Types of Defensibility-Related Requirements



Safety/Security Evidence Assurance Level (SEAL)

SEALs must be Well-Defined.

High SEALs require More Rigorous Development Method (including better Requirements Engineering):

- Formal Specification of Requirements
- Fagan Inspections of Requirements

Often SEALs only apply to Design, Coding, and Testing:

- Safe Subset of Programming Language
- Design Inspections
- Extra Testing (Test Types and Test Completion Criteria)

Architecture and Training (Methods) are also important.

Example Safety-Significant Requirements

Firing Missiles from Military Aircraft Requirements:

- When to Arm Missiles
- Controlling Doors before and after firing missiles
- Detecting Weight-On-Wheels

Chemical Plant Requirements:

- Mixing and Heating Chemicals
- Controlling Exothermic Reactions
- Detecting Temperature and Pressure

Example Types of Security-Significant Requirements

Access Control Requirements:

- Identification, Authorization, and Authorization

Integrity:

- Storage and Transmission of Sensitive Data
- Software that might get Infected by Malware
- Software that might represent Intellectual Property

Confidentiality:

- Handling of Sensitive Information

Availability (under attack):

- Services Subject to Denial-of-Services Attacks

Non-repudiation:

- Transactions

Safety and Security Function/Subsystem Rqmts.

Defensibility Function/Subsystem Requirements are requirements for functions or subfunctions that exist strictly to improve defensibility (as opposed to support the primary mission requirements).

- **Safety Function/Subsystem Requirements** are requirements for safety functions or subsystems.
- **Security Function/Subsystem Requirements** are requirements for security functions or subsystems.

Example Safety Function/Subsystem Rqmt. Types

Functions or subsystems strictly added for safety:

- Aircraft Safety Subsystems:
  - Collision Avoidance System
  - Engine Fire Detection and Suppression
  - Ground Proximity Warning System (GPWS)
  - Minimum Safe Altitude Warning (MSAW)
  - Wind Shear Alert
- Nuclear Power Plant:
  - Emergency Core Coolant System

All requirements for such functions/subsystems are safety-related.

Example Safety Function/Subsystem Rqmts

"Except when the weapons bay doors are open or have been open within the previous 30 seconds, the weapons bay cooling subsystem shall maintain the temperature of the weapons bay below X° C."

"The Fire Detection and Suppression Subsystem (FDSS) shall detect smoke above X ppm in the weapons bay within 2 seconds at least 99.9% of the time."

"The FDSS shall detect temperatures above X° C in the weapons bay within 2 seconds at least 99% of the time."

"Upon detection of smoke or excess temperature, the FDSS shall begin fire suppression within 1 second at least 99.9% of the time."

Example Security Function/Subsystem Rqmt Types

Functions or subsystems strictly added for security:

- Access Control
- Antivirus Subsystem
- Encryption/Decryption
- Firewalls
- Intrusion/Detection Subsystem

All requirements for such functions/subsystems are security-related.

Look in the Common Criteria (ISO/IEC 15408) for many reusable example security function requirements.

Example Security Function/Subsystem Rqmts

Access Control Function:

- The Access Control Function shall require users to identify themselves before enabling them to perform the following actions: …
- The Access Control Function shall require users to successfully authenticate their claimed identity before enabling them to perform the following actions: …
- The Access Control Function shall enable the system administrators to configure the maximum number of unsuccessful authentication attempts between the range of 1 and X.
- The Access Control Function shall perform the following actions when the maximum number of unsuccessful authentication attempts has been exceeded: …

Safety and Security Constraints

A **Constraint** is any Engineering Decision that has been chosen to be mandated as a Requirement. For example:

- Architecture Constraints
- Design Constraints
- Implementation Constraints
  (e.g., coding standards or safe language subset)
- Testing Constraints

A **safety constraint** is any constraint primarily intended to ensure a minimum level of safety (e.g., a mandated safeguard).

A **security constraint** is any constraint primarily intended to ensure a minimum level of security (e.g., a mandated countermeasure).

Safety and Security Standards often mandate Industry Best Practices as Constraints.

Example Safety/Security Constraints

Safety Constraints:

- The system shall use hardware interlocks to physically prevent users from coming into contact with moving parts.
- The system shall not have a single point of failure that can cause an accident.
- The system shall use a safe subset of C++.

Security Constraints:

- The system shall user use IDs and passwords for identification and authentication.
- The system shall incorporate firewalls to protect servers.
- The system shall incorporate a COTS virus detection product.
- The system shall use public key encryption to protect confidential information.
- The system shall use digital signatures to provide nonrepudiation.

Common Method:

A Basis for Effective Collaboration

You Are Here

Three Disciplines

Challenges

Requirements Engineering Overview

Safety and Security Engineering Overview

Types of Safety- and Security-related Requirements

Common Consistent Collaborative Method _

Conclusion

Desired Method Properties

Meet Challenges Listed at start of tutorial

Close Collaboration among Safety, Security, and Requirements Teams

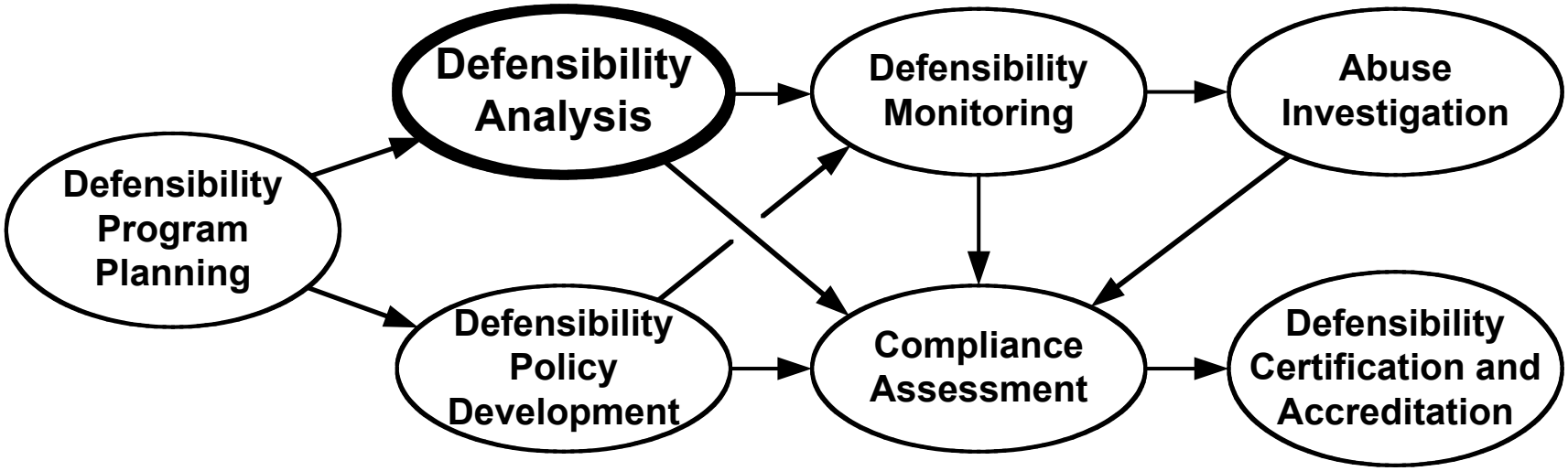
Better Integration of Safety and Security Methods:

- Based on Common Foundational Concepts and Terminology
- Reuse of Techniques and Work Products
- Based on Defensibility (Safety and Security) Analysis

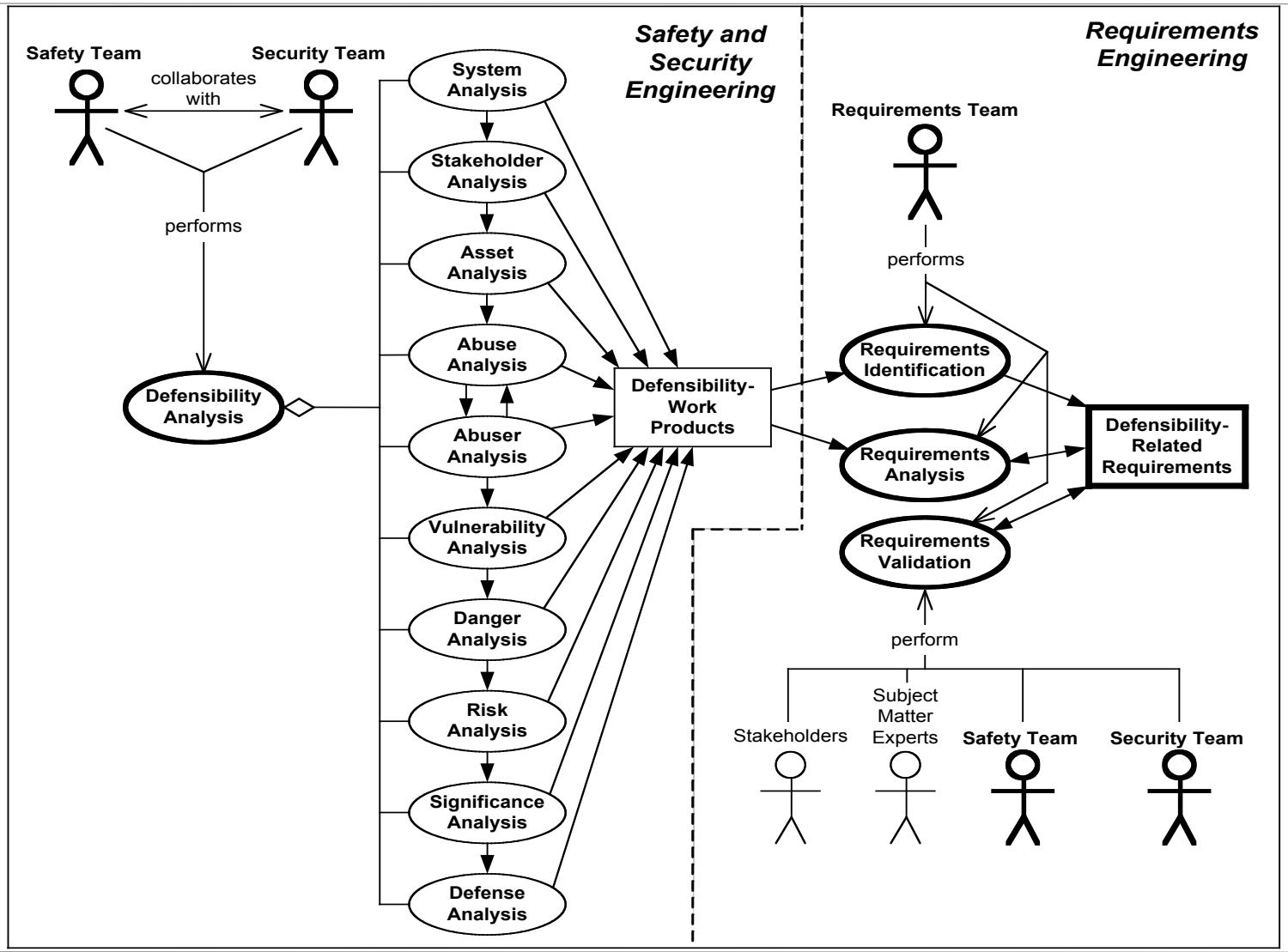
Better Integration of Safety and Security Engineering with Requirements Engineering:

- Clearly Defined Team Responsibilities
- Early Input to Requirements Engineering
- Develop all types of Safety- and Security-related Requirements
- Ensure these Requirements have the proper Characteristics

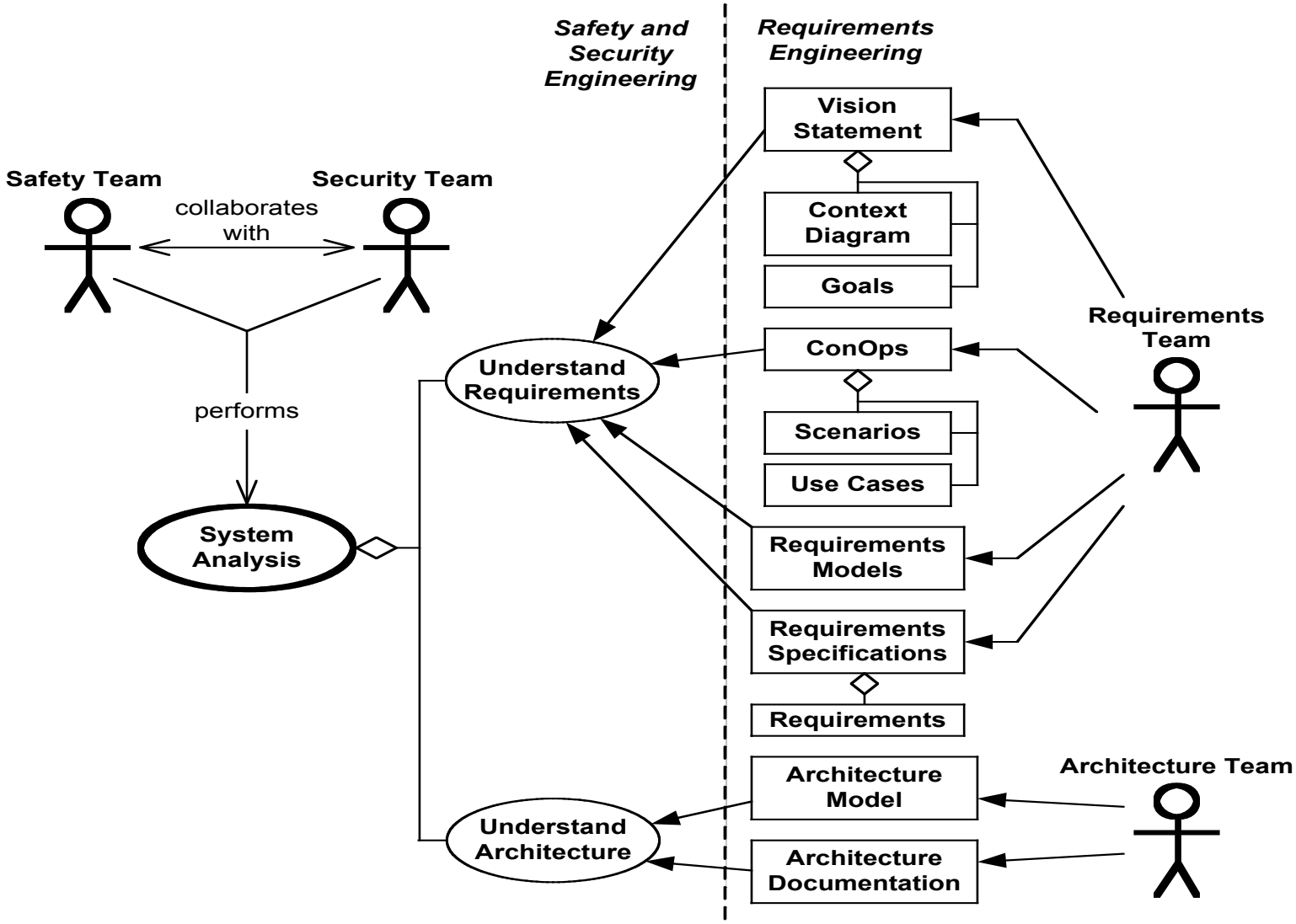
Overall Defensibility Engineering Method



Defensibility & Requirements Engineering



Systems Analysis



Common Example – Desired Characteristics

Common Ongoing Example throughout the Tutorial

Should Not Need Special Domain Knowledge

Example System should be:

- Safety-Critical
- Security-Critical
- Realistic
- SW-Intensive
- Understandable in terms of:
  - Goals and Requirements
  - Application Domain Technology
  - Accidents and Attacks
  - Safety Hazards and Security Threats

Example Overview

Very Large New Zoo

Zoo Automated Taxi System (ZATS)

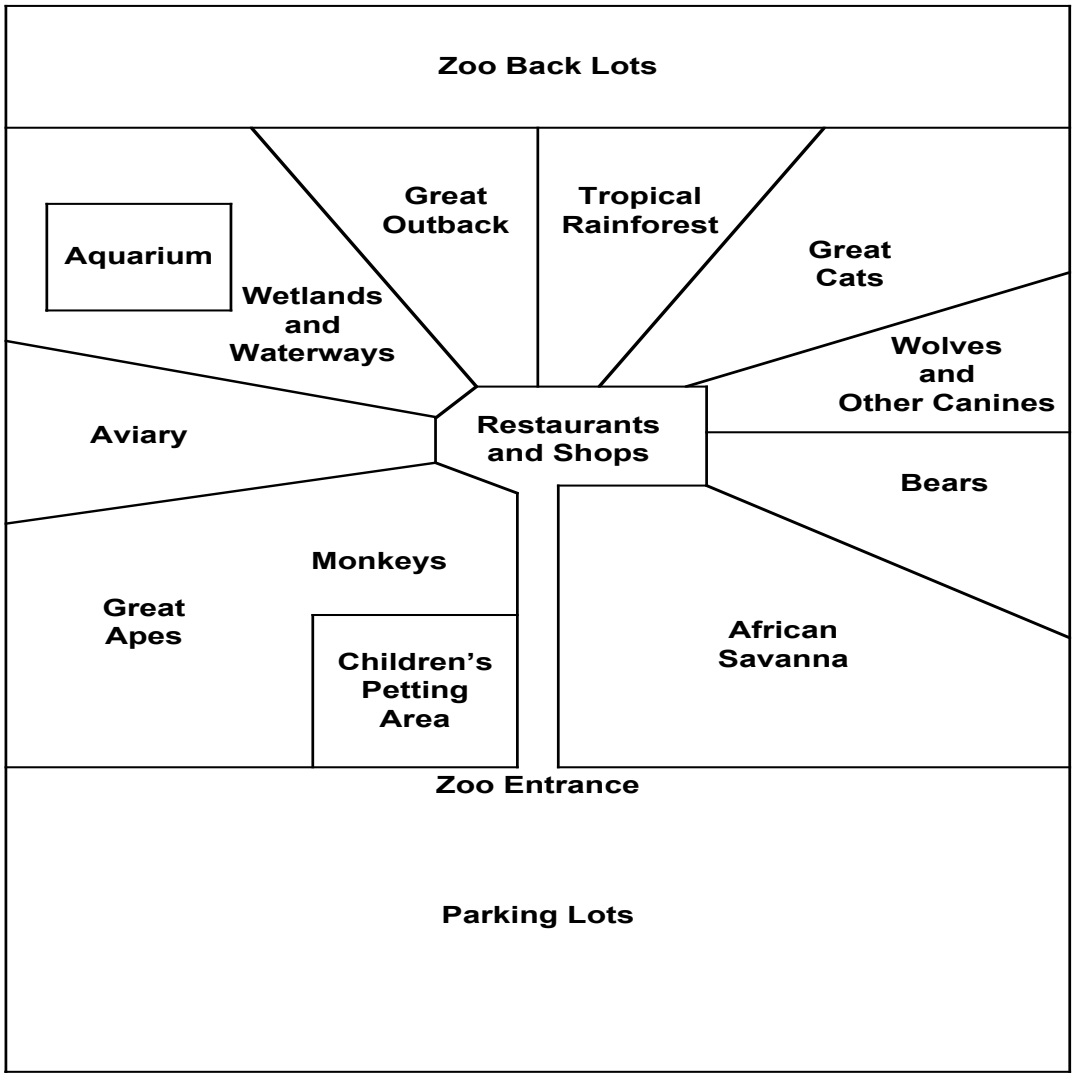
Example Zoo Habitat Guideway Layout

ZATS Context Diagram

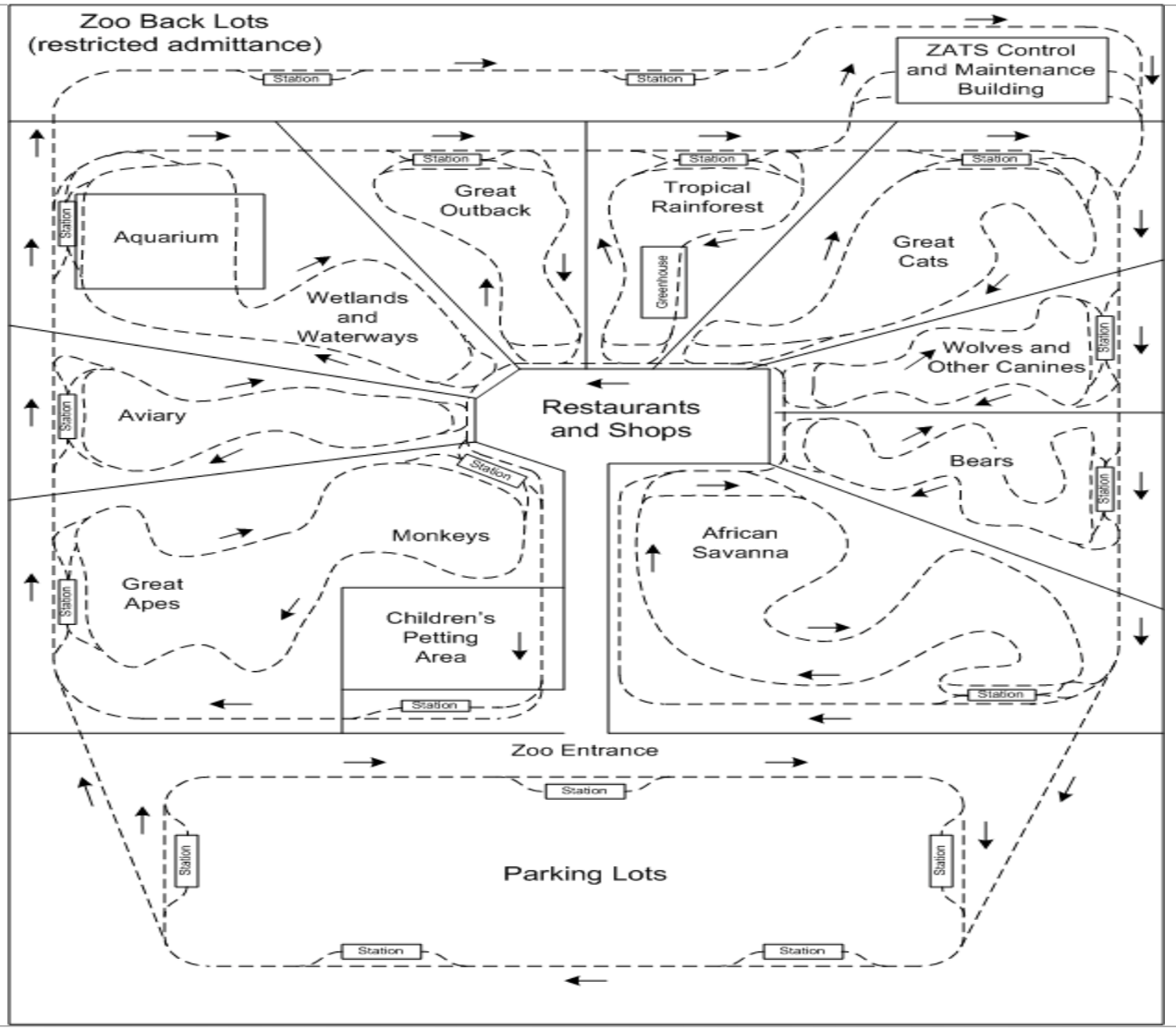
Proposed ZATS:

- Taxis
- Elevated Concrete Guideway
- Taxi Stations

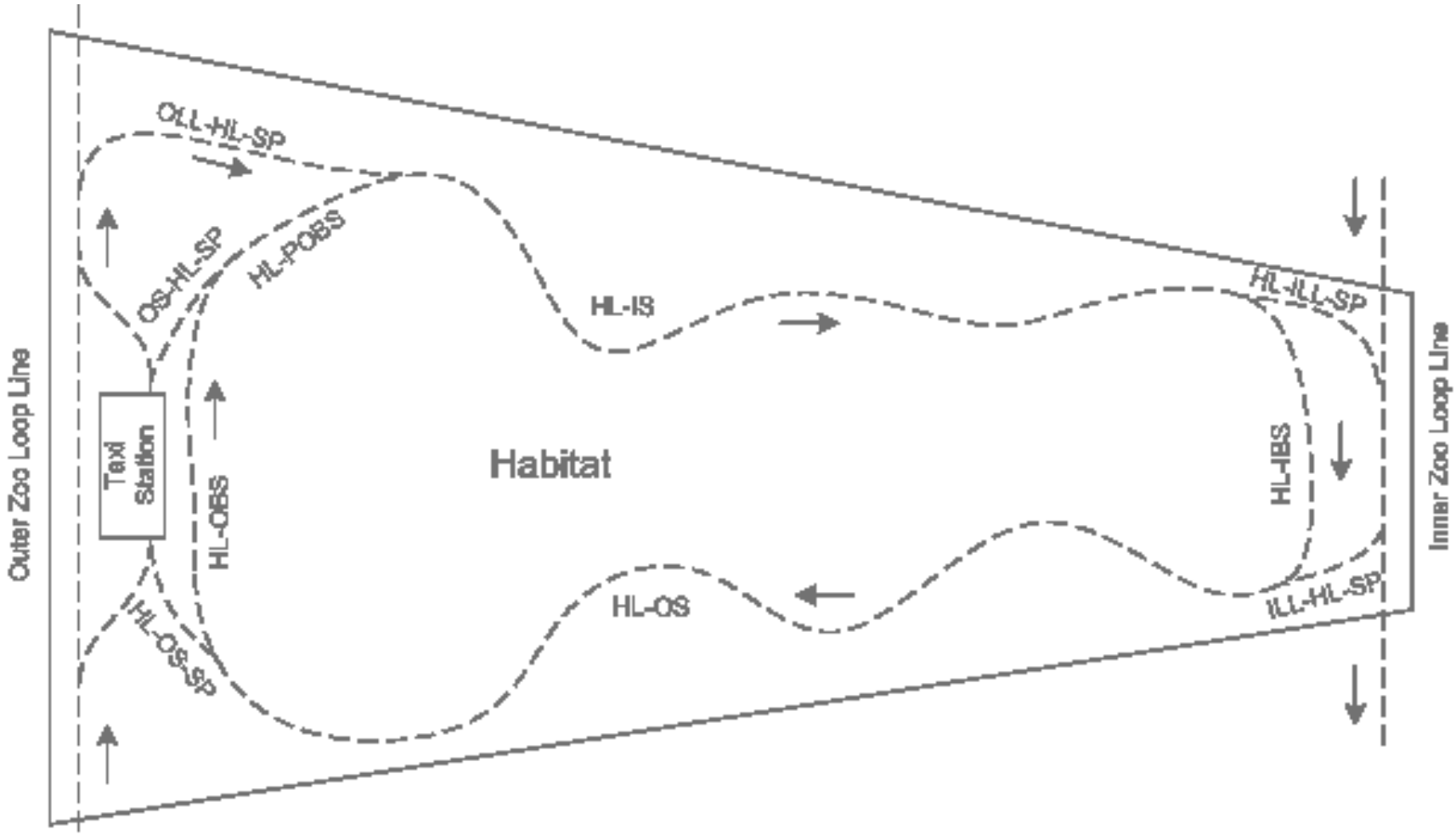
Very Large New Zoo



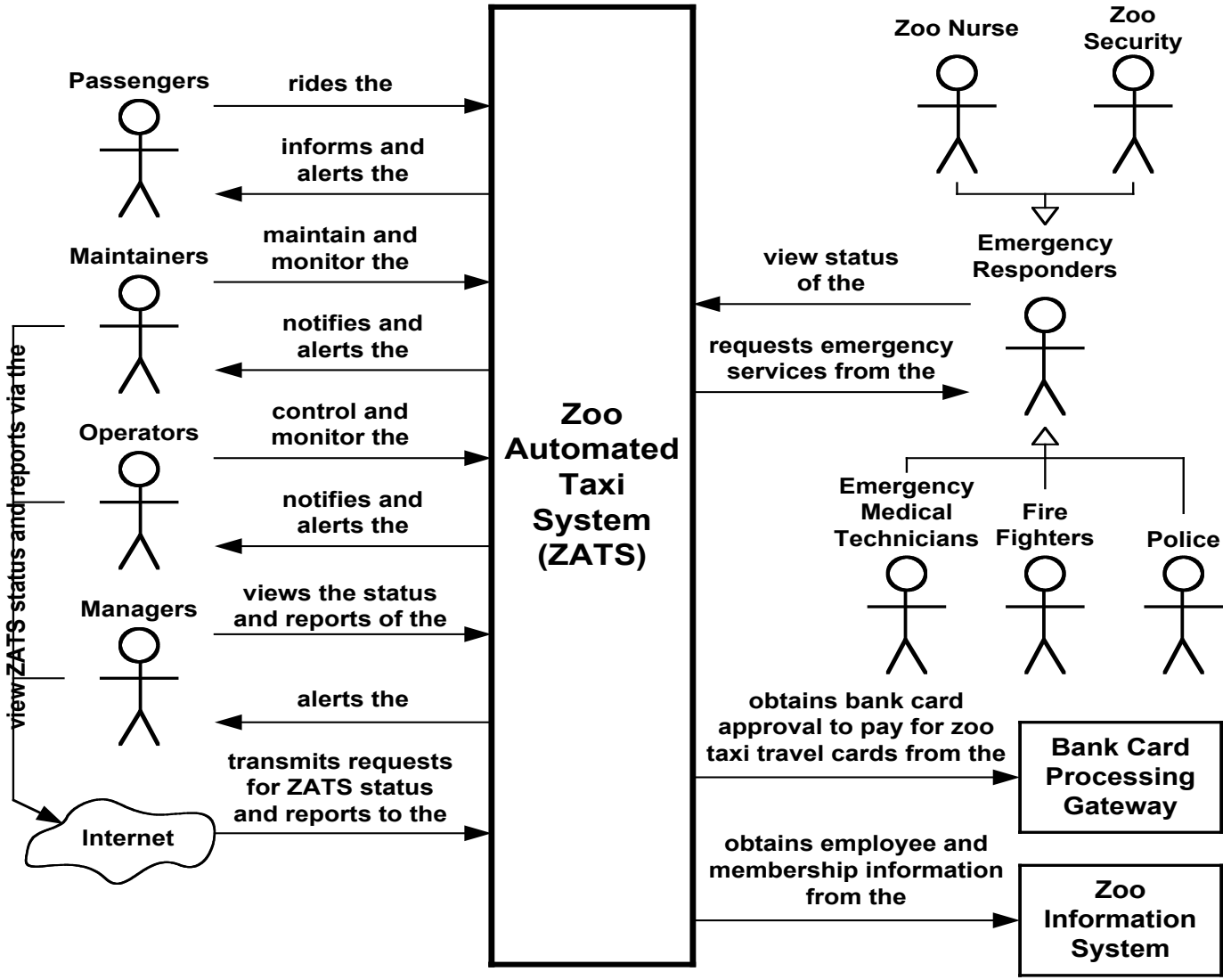
Zoo Automated Taxi System (ZATS)



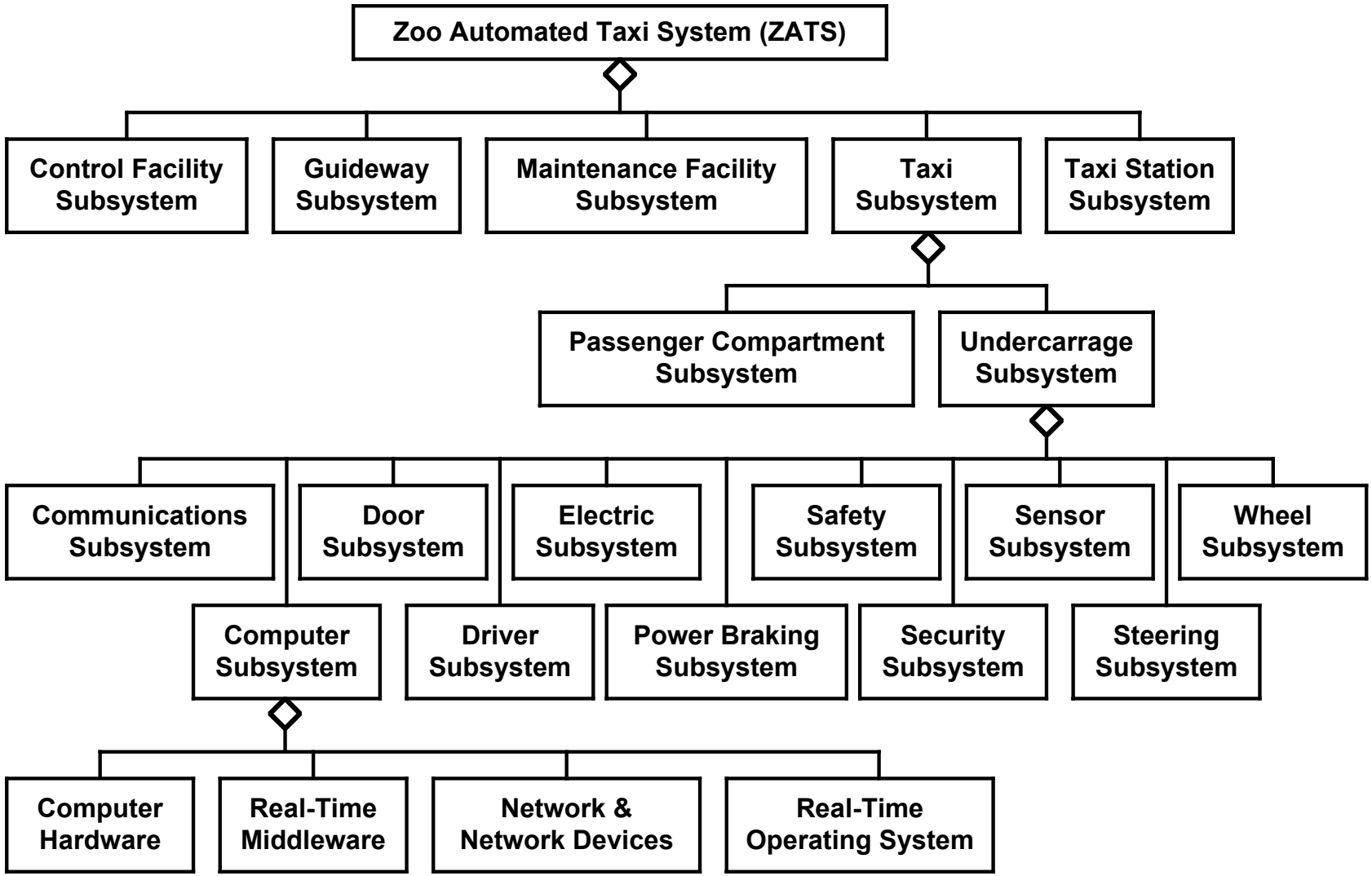
Example Habitat Layout



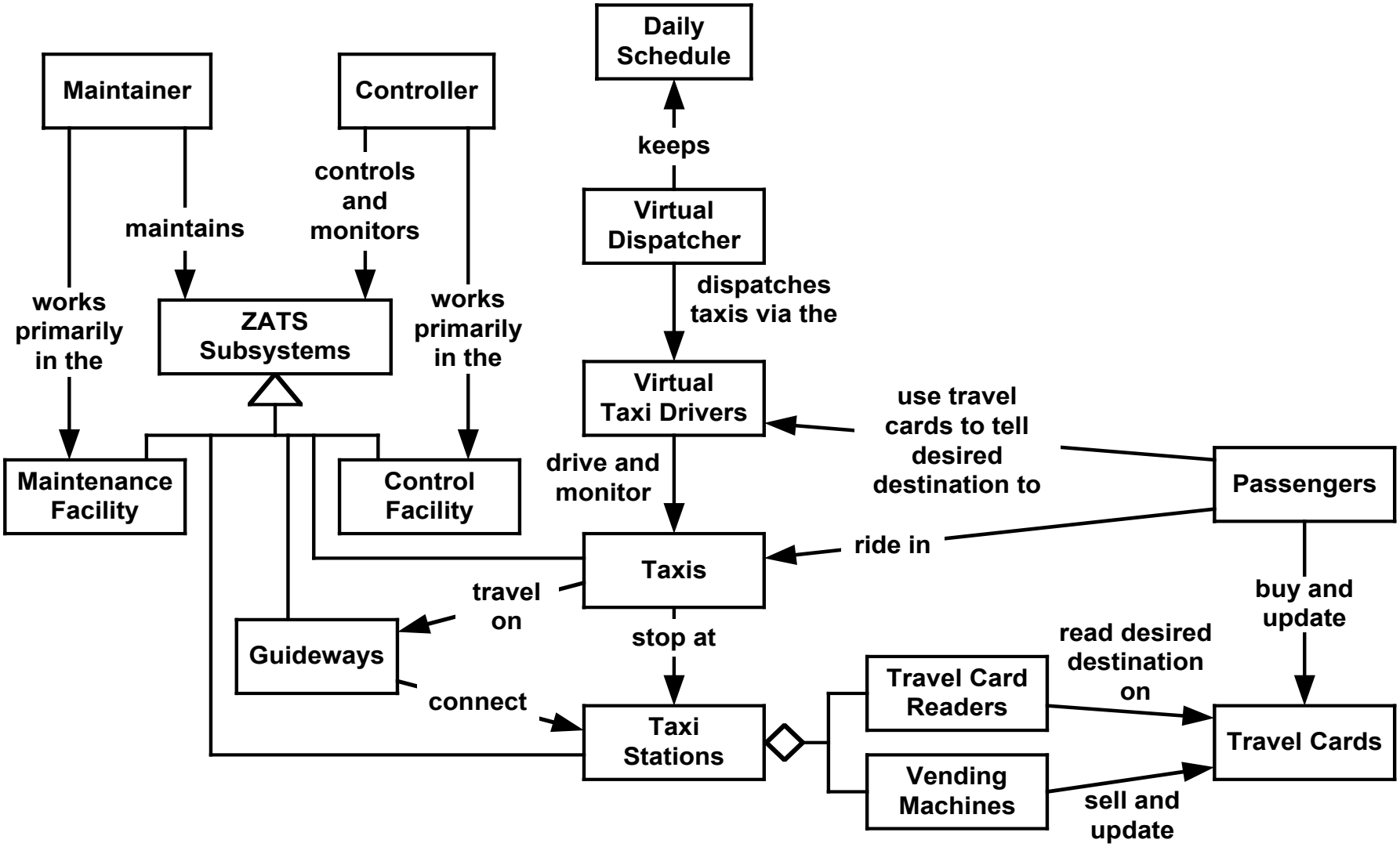
ZATS Context Diagram



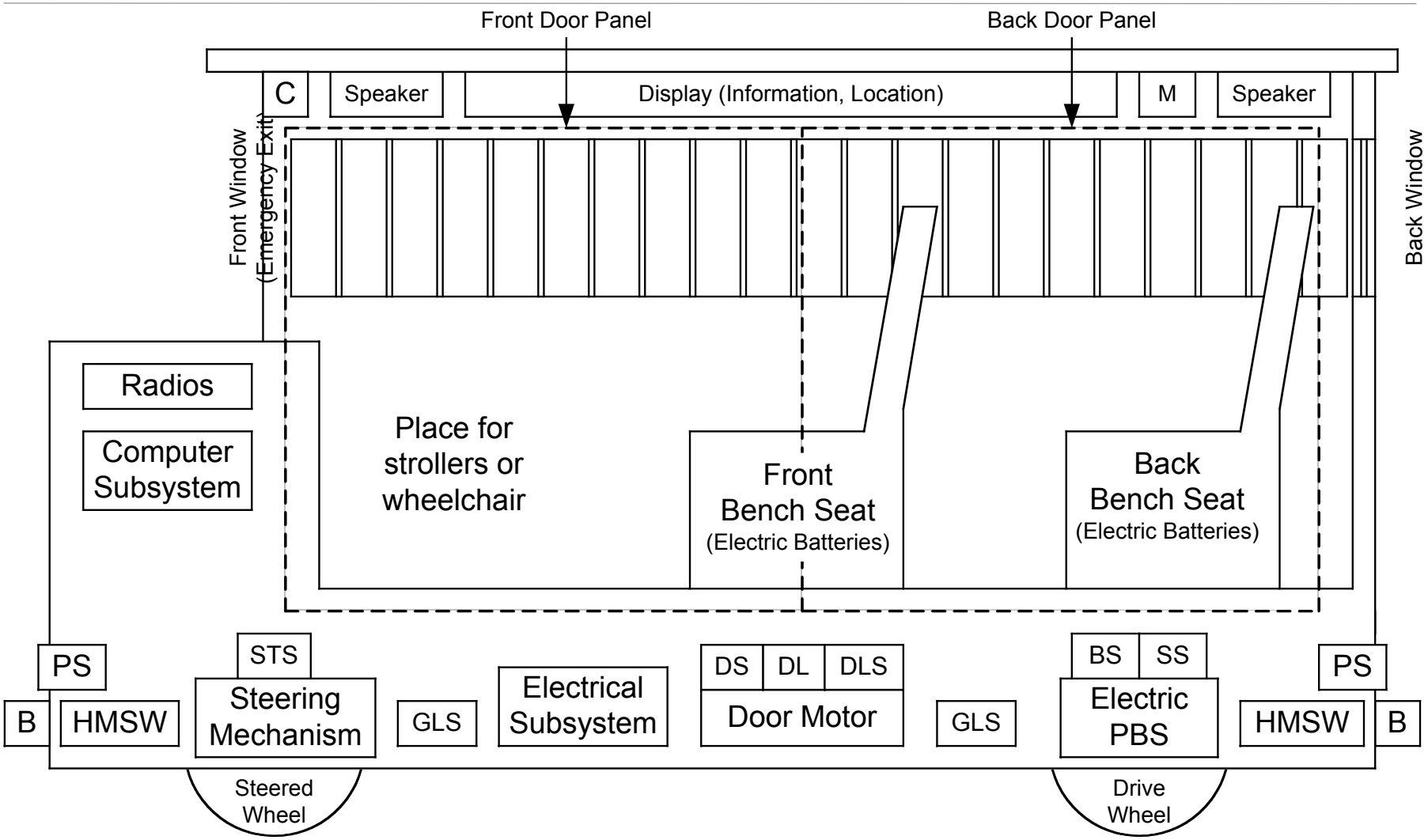
Proposed ZATS Decomposition (Partial)



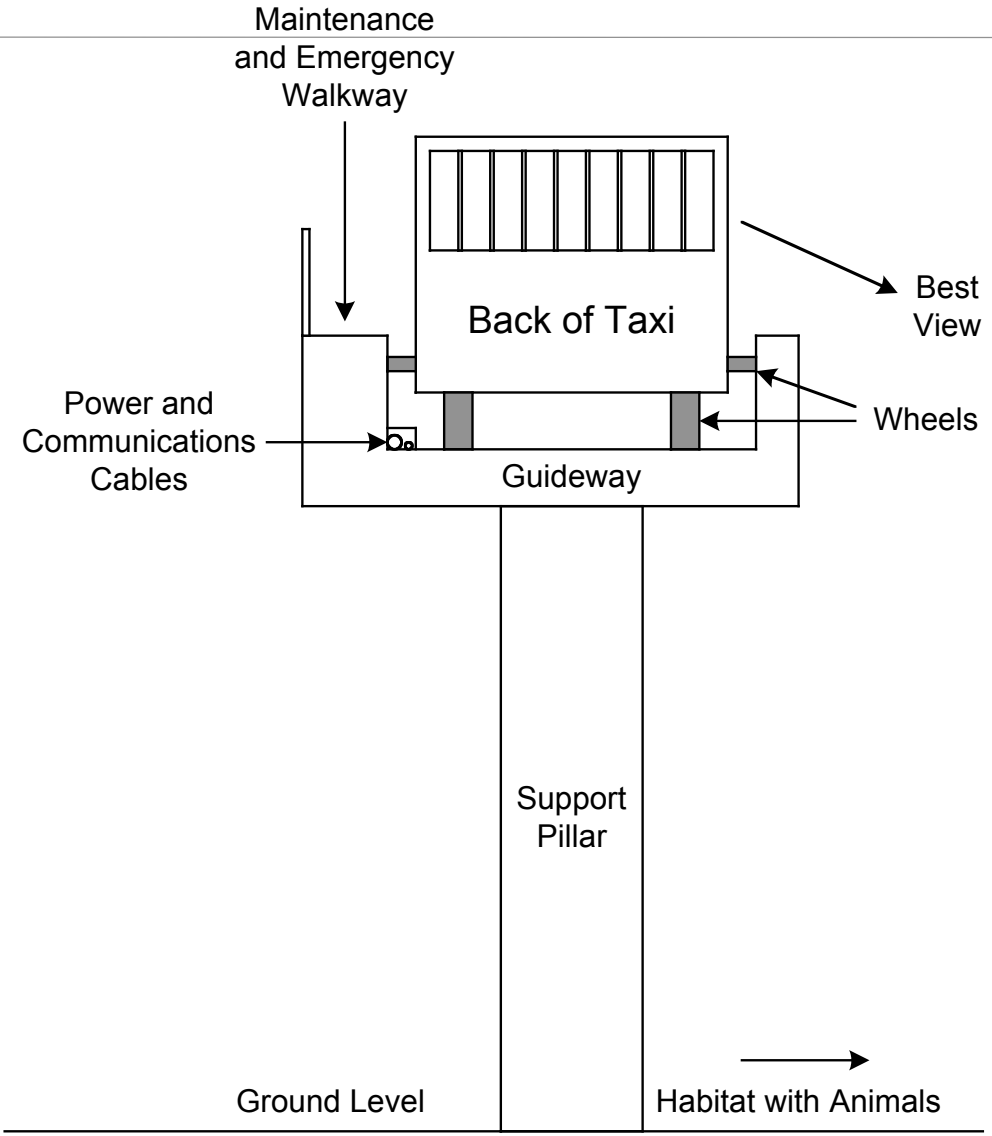
Proposed ZATS Domain Model



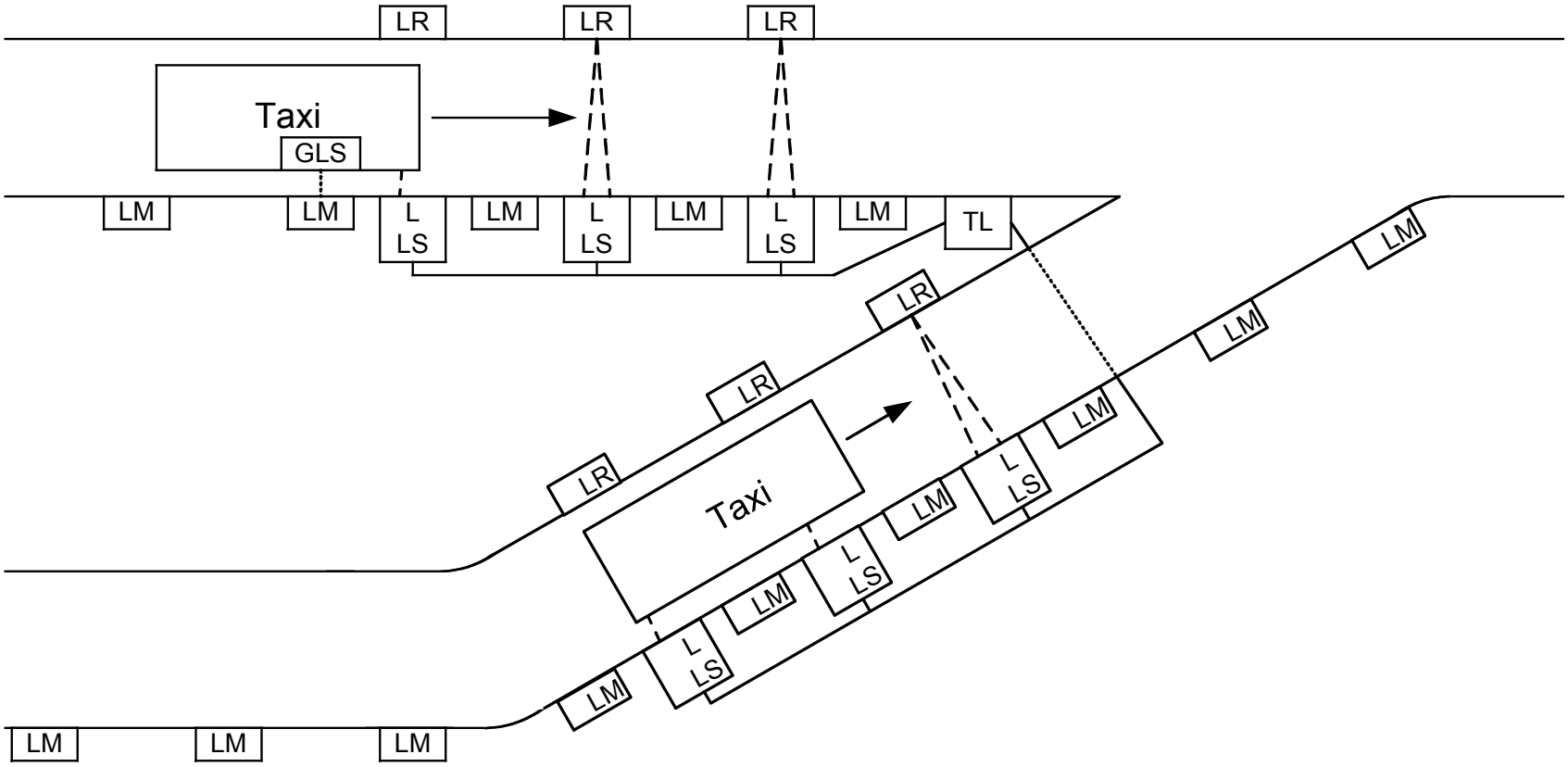
Proposed Taxi Architecture



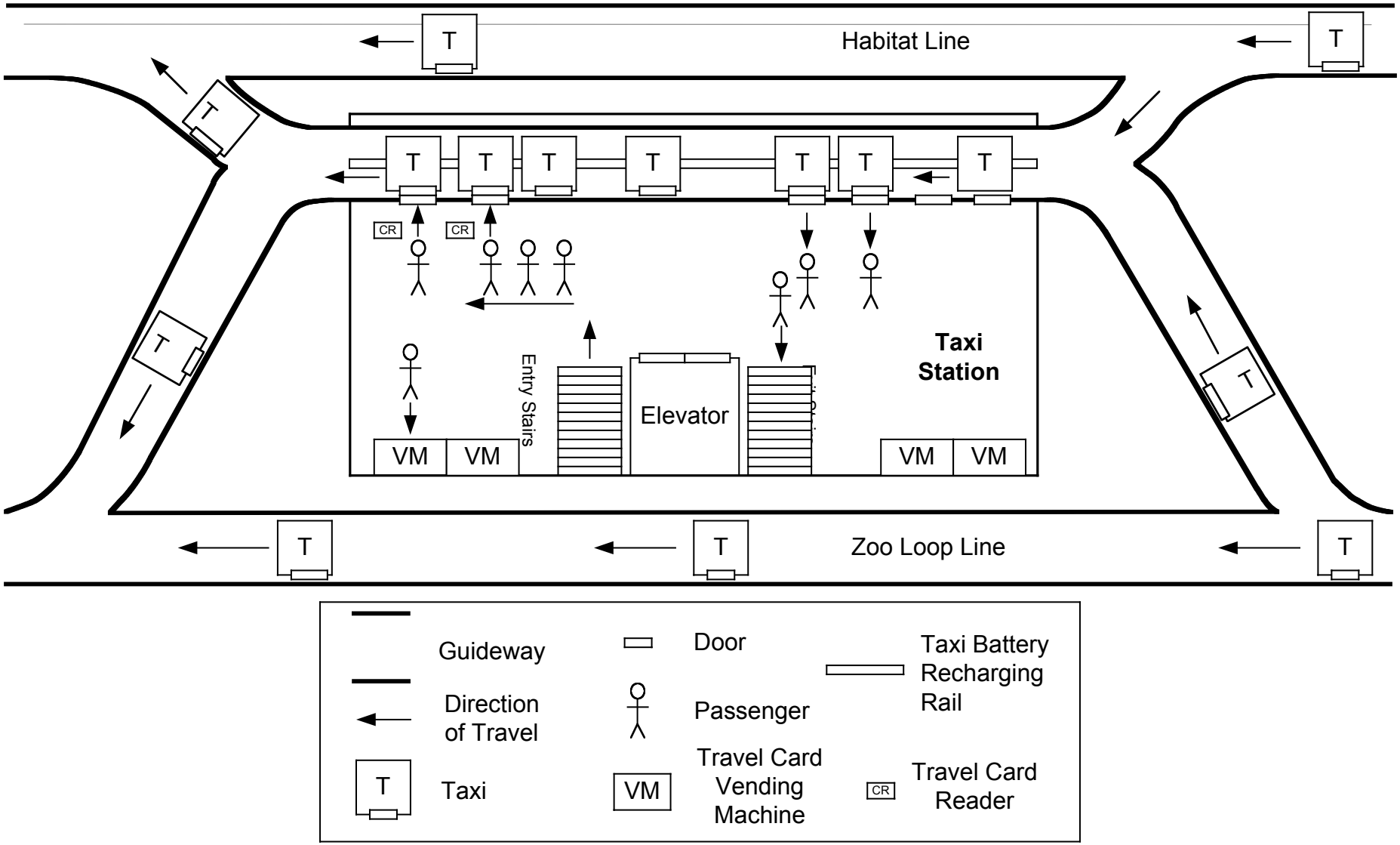
Automated Taxis On Elevated Guideways



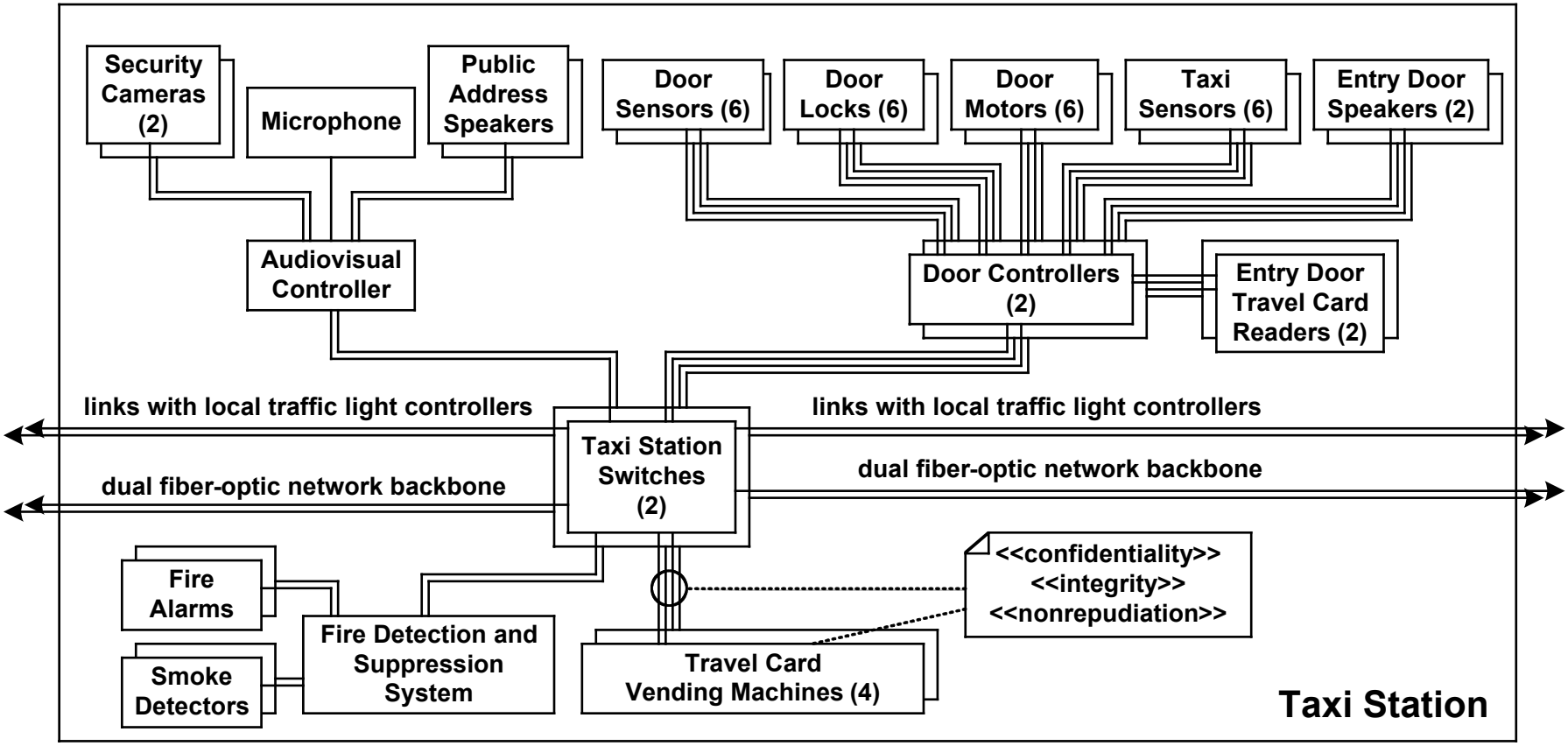
Example Collision Hazard



Proposed Taxi Station



Proposed Taxi Station Network Diagram



Example ZATS Goals

Functional Goals:

- ZATS must rapidly transport patrons between the parking lots and the zoo.
- ZATS must rapidly transport patrons between habitats within the zoo.
- ZATS must allow patrons to take leisurely tours of the habitats.

Data Goal:

- ZATS must record and report appropriate system usage statistics.

Capacity Goal:

- ZATS must include sufficient taxis so that patrons do not wait long for a taxi.

Usability Goal:

- ZATS must be very easy and intuitive for patrons to use, including those who are not good with technology.

Example ZATS Defensibility Goals

Safety Goals:

- "ZATS taxis must be safe."
- "ZATS must not have any serious accidents."
- "ZATS taxis must never collide."
- "ZATS will never kill or injure its passengers or maintainers."

Security Goals:

- "Passenger credit card data must be secure."
- "ZATS taxi service must be protected from denial of service attacks."
- "ZATS computers must prevent infection by malware."
- "ZATS facilities must be protected against arson."
- "ZATS must restrict users to only those tasks for which they are authorized."

Example ZATS Scenario

Ride Zoo Loop Line To Restaurants for Lunch:

After finishing their tour of the tropical rainforest habitat, the Smith family exit their taxi. They decide its time for lunch and walk over to look at the zoo map over the travel card vending machines. They decide to ride to the Great Apes and Monkeys taxi station near the central shops and restaurants area. Mr. Smith then swipes his zoo taxi travel card, and the display shows the remaining balance of $9.00 on the card. He selects the desired taxi station and then walks over to stand in line. He swipes his card through the card reader, and they enter the taxi. The taxi warns them to set down and thirty seconds later, the station and taxi exit doors close. Their taxi accelerates out of the taxi station and turns to the left onto the Zoo Loop Line.

Shortly after leaving the taxi station, they see a spur the angles off to their left towards a large building containing the taxi control center and maintenance facility. They continue around the outside of the zoo, passing other the Great Cats, the Wolves and Other Dogs, and the Bears habitats. Just before they reach the outer African Savanna taxi station, the guideway makes a sweeping turn to the right and they can see the parking lot on their left. Everyone looks to see if they can see the family van, but the parking lot is too big and they can only see the parking lot taxi station near where it is parked.

Soon, they pass the zoo entrance on their left and turn right to follow the main street to where the main restaurants and shops are. Their taxi passes the inner African Savanna taxi station on their right, circles around the central area, and soon pulls off the Zoo Loop Line to enter the inner Great Apes and Monkeys taxi station. Exiting the taxi when the doors open, they head down the elevator and outside for an early lunch at one of the many restaurants.

Representative ZATS Functional Requirement

Prepare for departure warning:

- When a taxi is in the IN SERVICE – STOPPED AT STATION state and its passengers have selected a tour of a habitat and paid for the tour, then (1) ZATS *shall* warn the passengers in the taxi and in the taxi station in front of the taxi (a) to stop boarding that particular taxi because the doors will soon close and (b) to stay away from the doors and (2) ZATS shall transition the taxi to the IN SERVICE – PREPARE FOR DEPARTURE state.

Note precondition, trigger, required behavior, and postcondition.

Representative ZATS Data Requirement

ZATS shall record the following information about each trip:

- Taxi Identifier
- Taxi Travel Card:
  - Identifier
  - Debit Amount
  - Remaining Balance
- Starting Station
- Destination Station
- Departure Time
- Arrival Time

Representative ZATS Interface Requirements

ZATS shall interface with the Bank Card Processing Gateway in order to request bank card approval to pay for zoo taxi travel cards.

ZATS shall interface with the Zoo Information System to obtain employee and zoo membership information.

ZATS shall interface with the emergency responder systems (e.g., 911) to enable the:

- Operator to request emergency services
- Emergency responders to view ZATS status

Representative ZATS Constraints

Architecture Constraints:

- ZATS shall use pre-stressed reinforced concrete guideways able to support 150% max. expected loading.
- ZATS guideways shall be elevated to provide good visibility, to separate patrons from the animals, and to eliminate the possibility of collision between taxis and patrons' vehicles in the parking lots.
- ZATS shall use COTS electric motors.
- ZATS taxis shall use standard automobile tires.
- ZATS shall use a commercial real-time operating system.

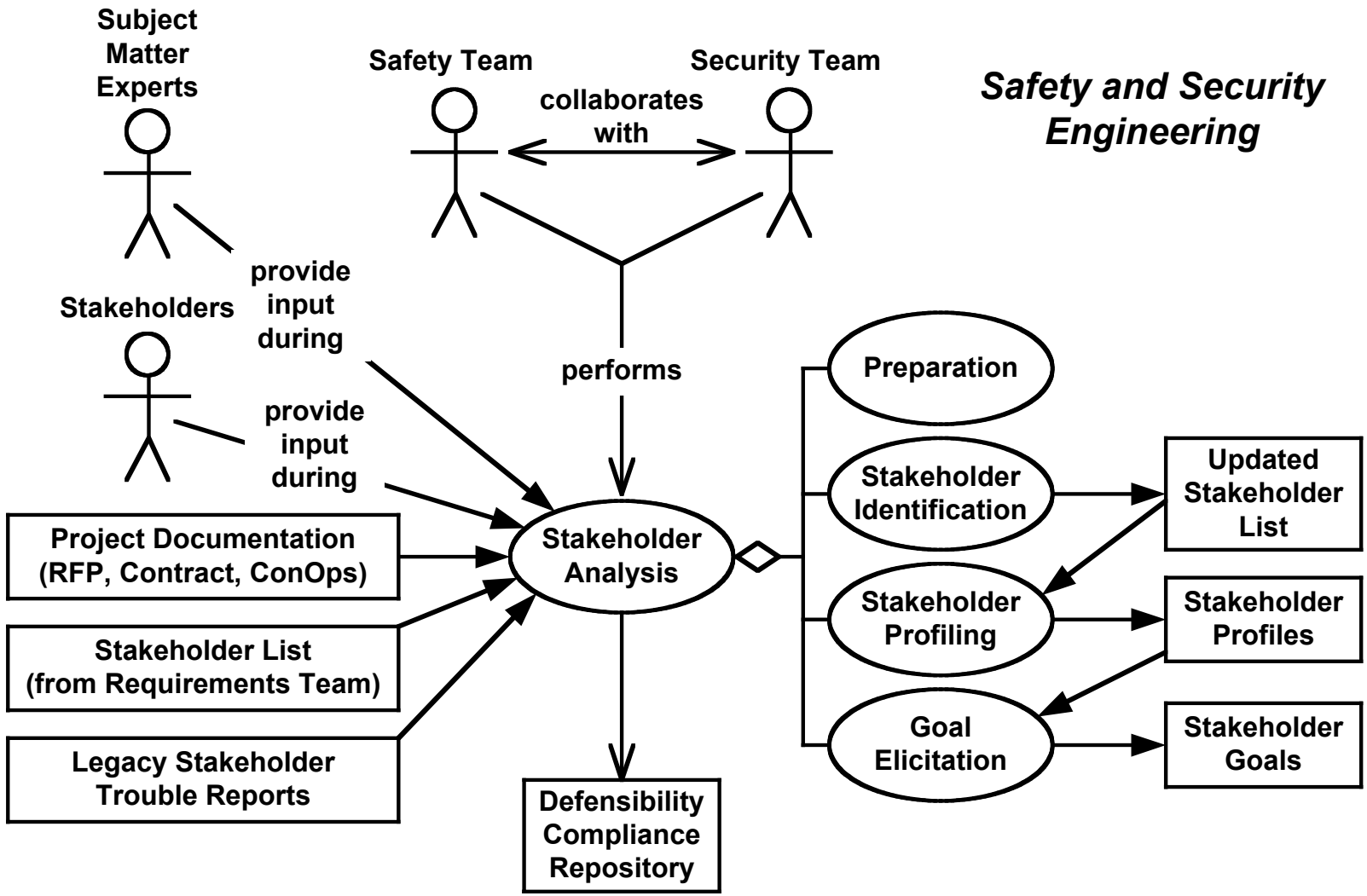
Design Constraints:

- ZATS software shall be object-oriented.

Implementation Constraints:

- ZATS software shall be programmed in a safe subset of C++.

Stakeholder Analysis



Example ZATS Stakeholders

People:

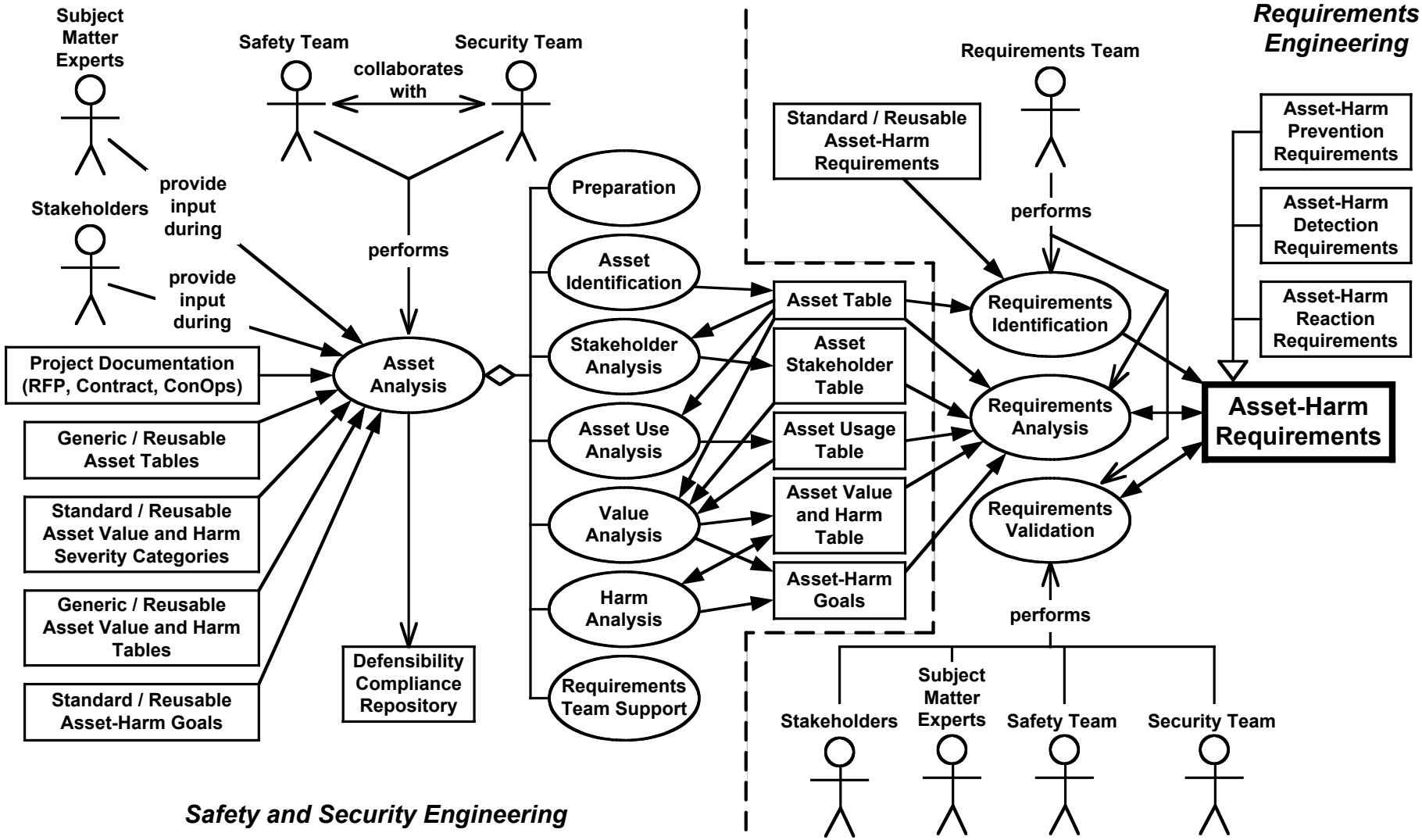
- Emergency Responders
- Passengers
- Operators
- Maintainers
- ZATS Developers
- Zoo Employees
- Zoo Management

Organizations:

- Bank Card Processing Gateway
- Safety and Security Certification/Accreditation Bodies
- Zoo Regulatory Bodies

Zoo Animals (Stakeholder or Valuable Asset?)

Asset Analysis



Example ZATS Valuable Assets

People:

- Passengers
- Operators
- Maintainers

Property:

- Animals
- Money
- Passenger Bank Card Information
- Reputation
- Taxis
- Taxi Stations

Environment:

- Habitat

Services:

- Taxi Service

Example ZATS Assets Value Categories

Priceless:

- Incalculable value far beyond that of the system
  (person, endangered animal, over $1,000,000, and water table under zoo)

Extreme:

- Value far above the average

High:

- Value significantly above the average

Moderate:

- Value within reasonable or average limits

Low:

- Value well below average and of little consequence

Definitions must be operationalized in terms of example people, property, the environment, and services.

Example ZATS Harm Severity Categories

Catastrophic:

- Potential ZATS lifespan harm that is unacceptable to authoritative stakeholders

Major:

- Potential five-year harm that is only acceptable to authoritative stakeholders after major actions have been taken to lower its risk

Minor:

- Potential yearly harm that is acceptable to authoritative stakeholders after minor action has been taken to lower its risk

Negligible:

- Potential yearly harm that is acceptable to authoritative stakeholders but that does not justify any action to lower its risk

Definitions must be operationalized in terms of harm types and levels.

Example ZATS Asset-Harm Goals

Safety Asset-Harm Goals:

- ZATS will not harm any passengers.
- ZATS will not harm any animals.
- ZATS will not damage any of its components.
- ZATS will protect itself from accidental fires.

Security Asset-Harm Goals:

- ZATS will protect its passenger's bank card information.
- ZATS will protect the money used to purchase tickets.
- ZATS will protect itself from arson.

Example ZATS Asset-Harm Requirements

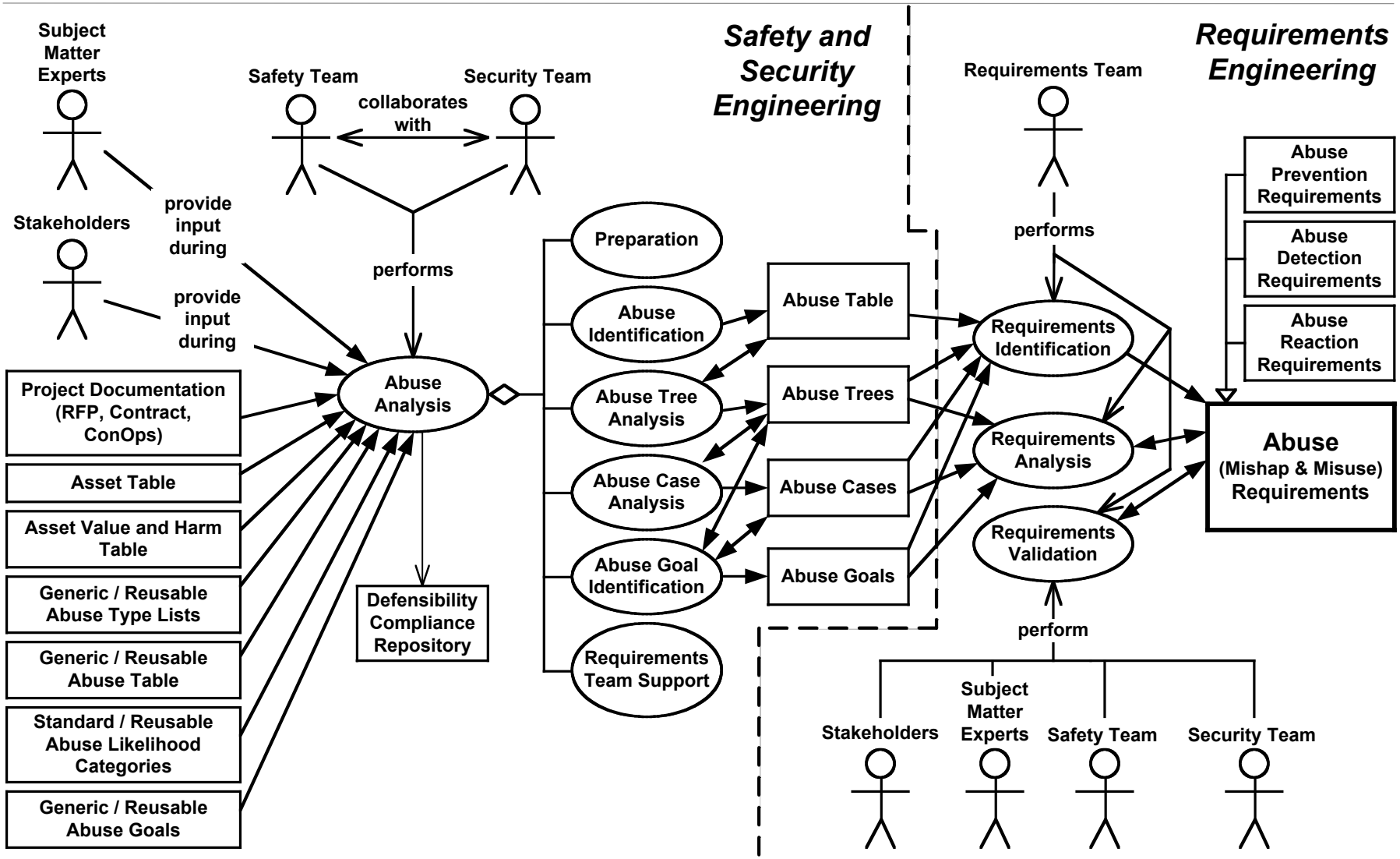
Safety Asset-Harm Requirements:

- ZATS shall not kill any passengers with a probability exceeding 0.1 passenger per expected 30 system lifespan.
- ZATS shall prevent the total destruction of taxi stations due to accidental fire with a probability of no more than one such fire per expected 30 system lifespan.

Security Asset-Harm Requirements:

- ZATS shall protect its passenger's confidential bank card information by ensuring that the expect of unauthorized malicious access to this information to less than

Abuse (Misuse and Mishap) Analysis



Example Types of ZATS Safety Abuses (Mishaps)

Accidents:

- Natural Disasters
- Taxi Accidents
- Taxi Station Accidents

Safety Incidents:

- Inadequate Headway
- Overspeed

Example Types of ZATS Security Abuses (Misuses)

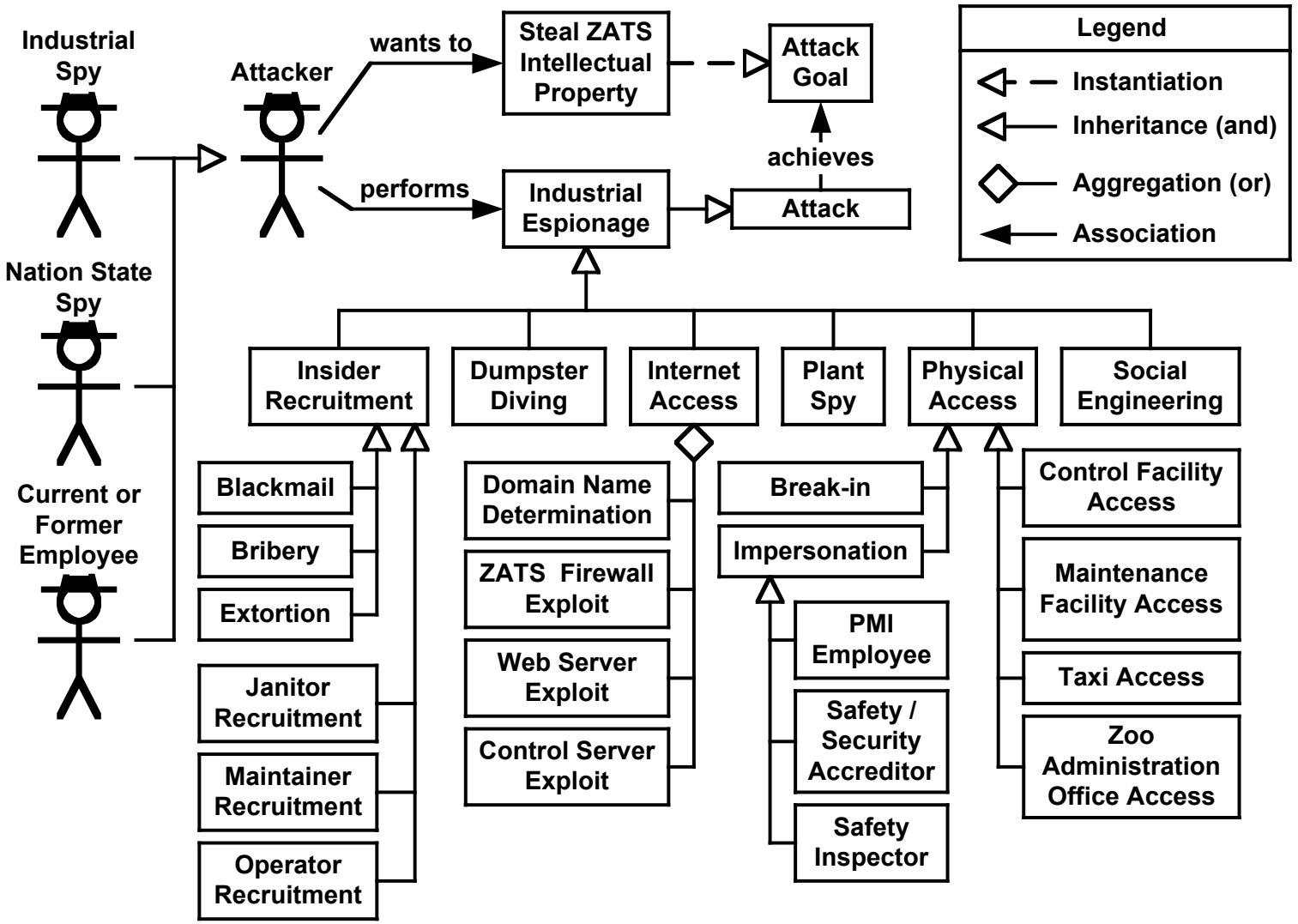
Attacks:

- Arson
- Cyber-Attacks
  - Denial of Service Attacks
  - Malware Attacks
  - Theft of Bank Card Data
- Muggings
- Theft of Equipment

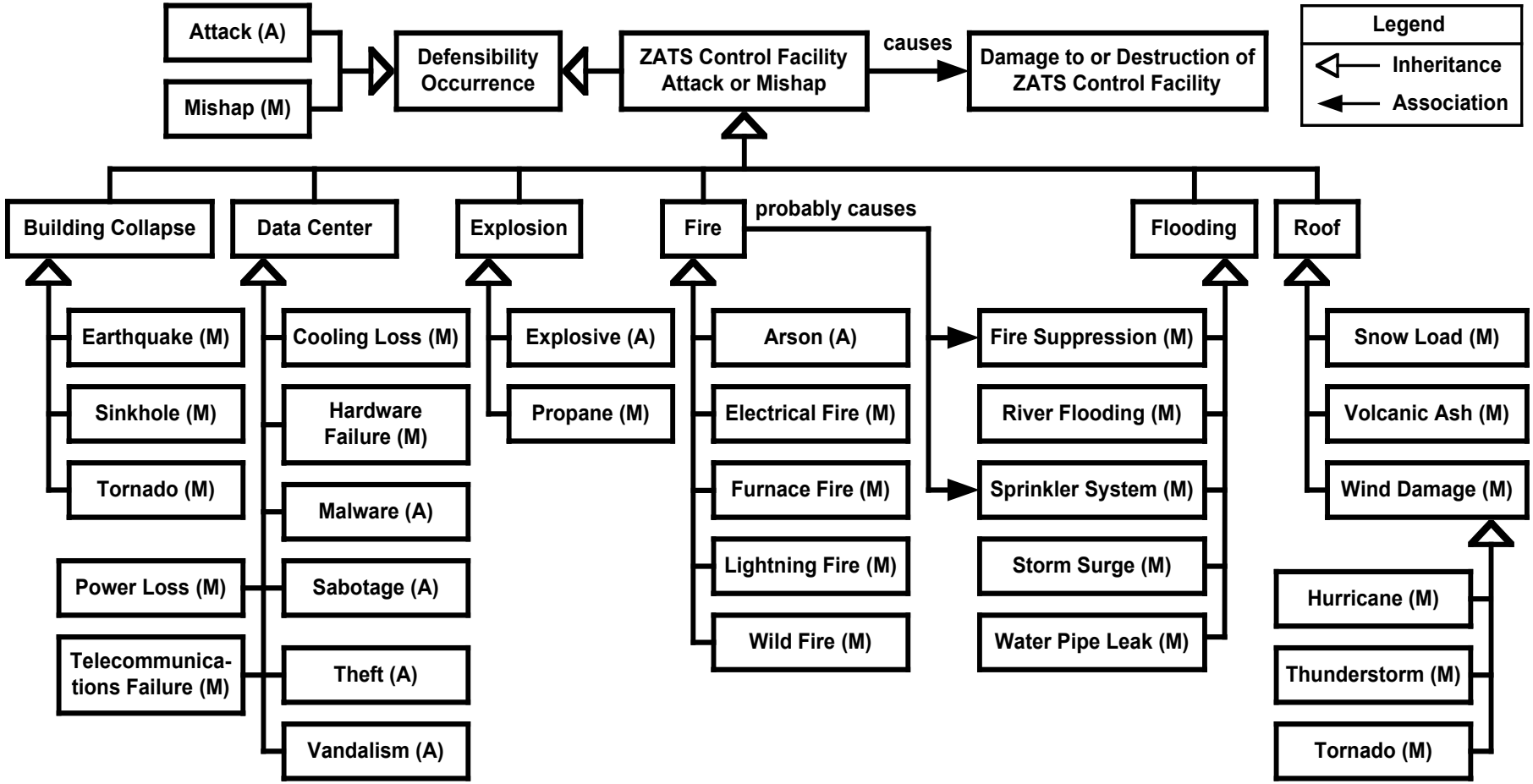
Security Incidents:

- Security Probes

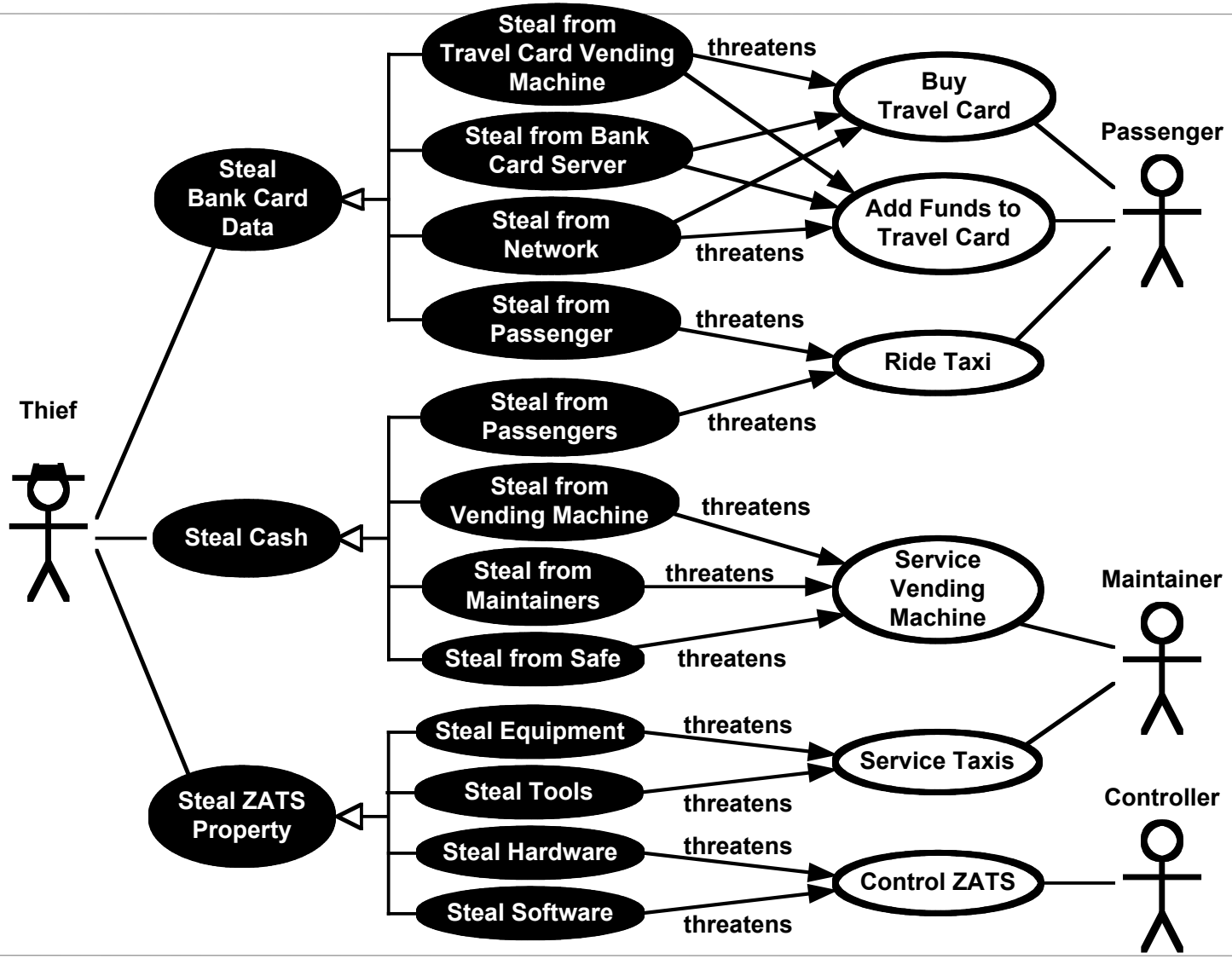
Example Abuse (Attack) Tree



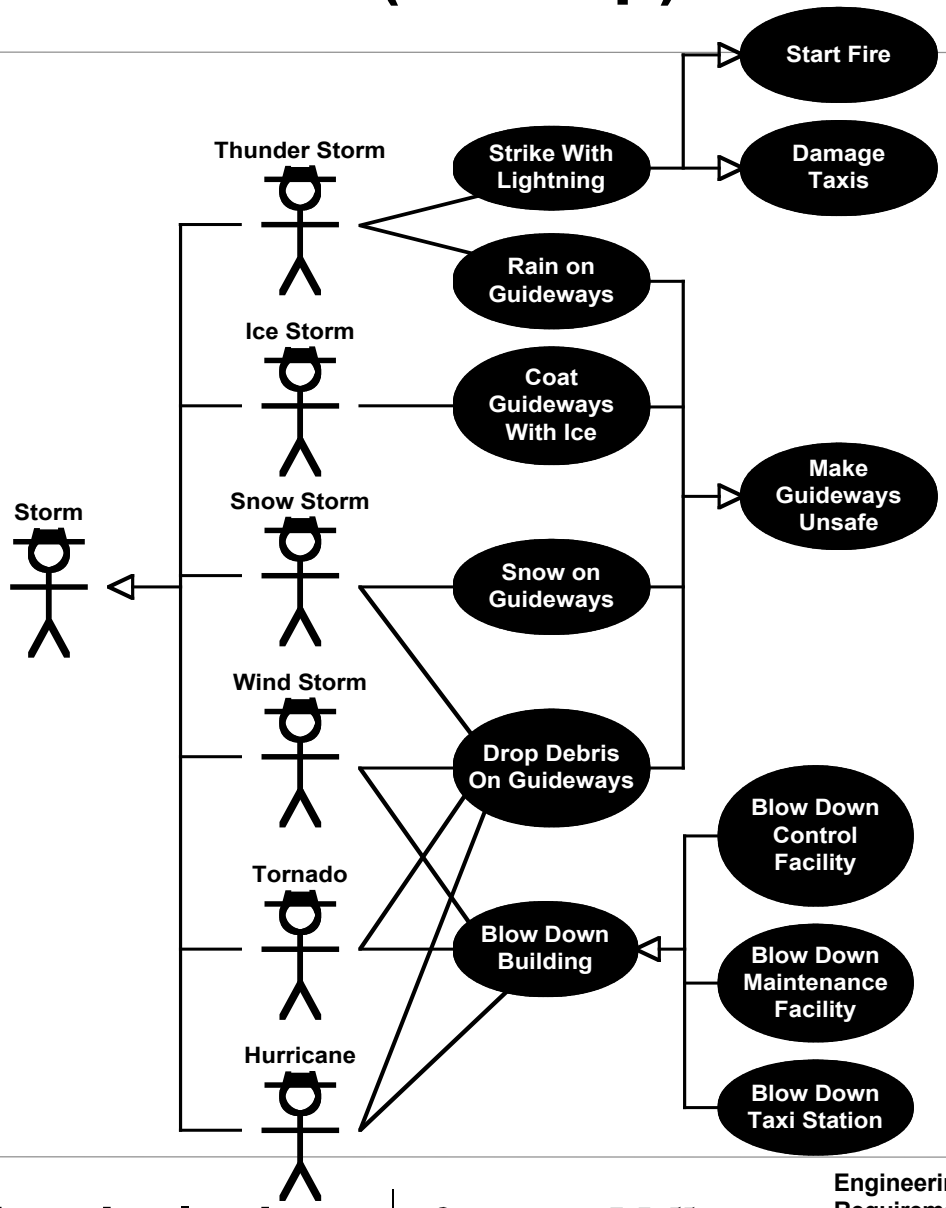
Example Abuse (Mishap and Attack) Tree



Example ZATS Abuse (Misuse) Cases



Example ZATS Abuse (Mishap) Cases



Example ZATS Abuse Goals

Safety Mishap Goals:

- ZATS will survive natural disasters such as floods and hurricanes.
- ZATS will prevent all taxi accidents including collisions with both other taxis and the guideways.
- ZATS will prevent all taxi station accidents including accidents involving doors and elevators.

Security Misuse Goals:

- ZATS taxi stations will resist arson attacks.
- ZATS will resist denial of service attacks.
- ZATS computers will resist malware
- ZATS will protect passenger bank card information.
- ZATS will protect its passengers from muggings.
- ZATS will protect its valuable equipment from theft.

Example ZATS Abuse (Mishap) Requirements

Prevention Requirements:

- **Taxi Collisions:** Under normal operating conditions, ZATS shall ensure that the rate of major taxi collisions* is less than X per [trip | time unit].
- **Taxi – Guideway Collisions:** Under normal operating conditions, ZATS shall ensure that the rate that taxis collide with guideways at junction points is less than X per [guideway junction | trip | time unit].

Detection Requirements:

- **Taxi Collisions:** ZATS shall detect major taxi collisions at least 99.9% of the time.

Reaction Requirements:

- **Taxi Collisions:** When ZATS detects a major taxi collision, then it shall notify the ZATS operator, the Zoo Nurses Office, and the Zoo administration office within 1 minute at least 99.99% of the time.

* Terms must be properly defined. For example, major taxi collision is one with a relative speed of at least 10 mph.

Example ZATS Abuse (Misuse) Requirements

Prevention Requirements:

- **Unauthorized Internet Access:** ZATS shall provide technical means (e.g., firewalls, proper switch configuration, and encryption) that are sufficient to prevent industrial spies with profile X from using the Internet to successfully obtain sensitive intellectual property for a minimum of Y [units of time].

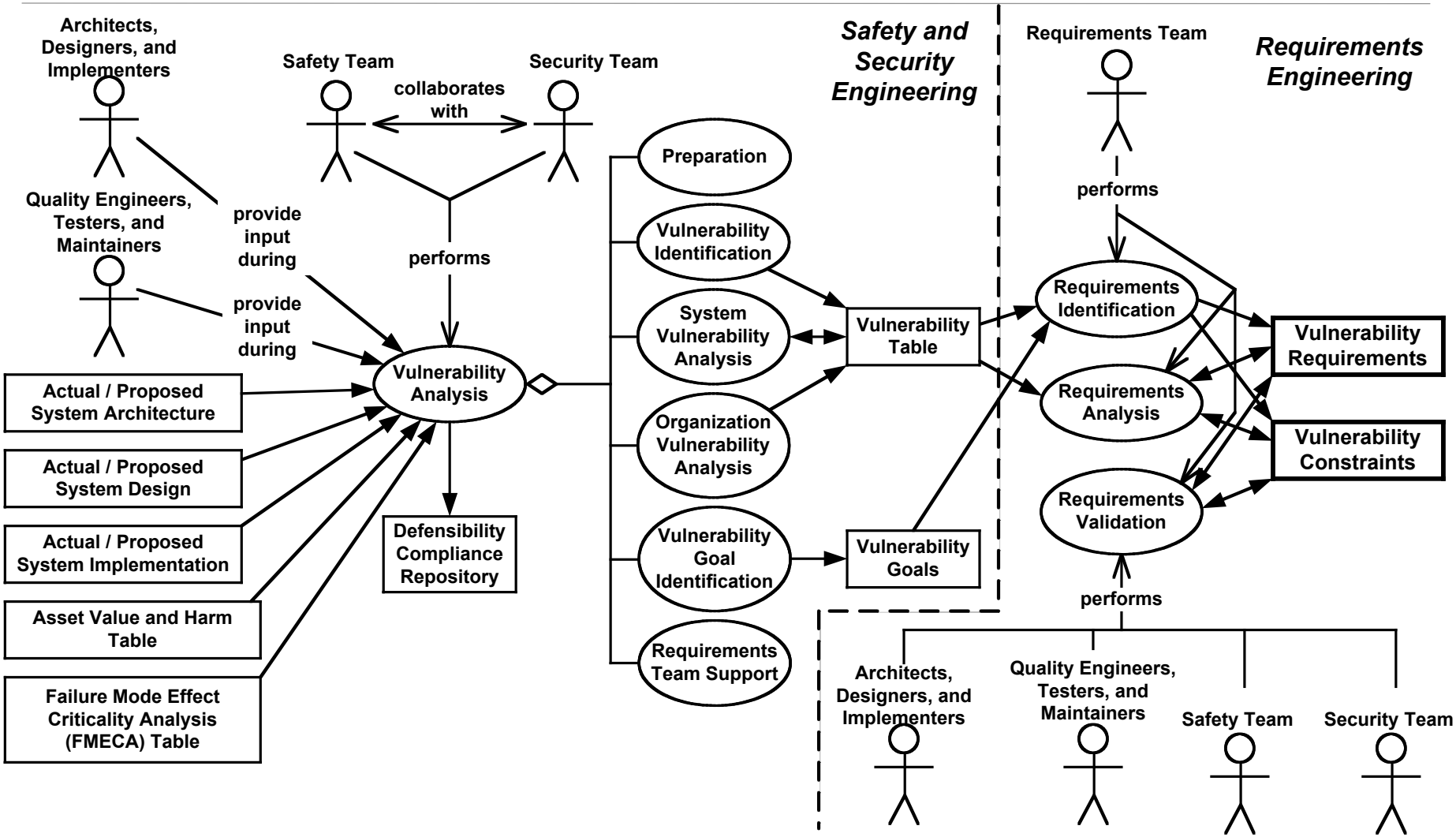
Detection Requirements:

- **Unauthorized Internet Access:** ZATS shall incorporate technical means (e.g., an intrusion detection system) that are sufficient to detect at least 99% of all unauthorized attempts at Internet access.

Reaction Requirements:

- **Unauthorized Physical Access:** If ZATS detects a break in at the ZATS control facility or maintenance facility, then ZATS shall notify the ZATS operator, the zoo security department, contact the police, and record the security event at least 99% of the time.

Vulnerability Analysis



Example ZATS Vulnerabilities

Safety Vulnerabilities (Potential/Actual):

- Hardware / Software Defects
  - Defect in Safety-Critical Hardware, Software, and Safeguards
- No Door Locks and Door Sensors
- No Fire Detection and Suppression System

Security Vulnerabilities (Potential/Actual):

- Hardware / Software Defects
  - Defect in Security-Critical Hardware, Software, and Countermeasures
- Software contains Trapdoors
- Sensitive Bank Card Information Not Encrypted
- Poor, Missing, or Obsolete Virus Detection
- Poor or Missing Operator / Maintainer Identification, Authentication, and Authorization
- Missing or improperly Configured Firewalls

Example ZATS Vulnerability Goals

Safety Vulnerability Goals:

- ZATS taxis will contain no safety vulnerabilities.
- ZATS guideways will contain no safety vulnerabilities.
- ZATS will have sufficient safeguards, which will be properly installed and maintained.

Security Vulnerability Goals:

- ZATS control facility, maintenance facility, and taxi stations will contain no security vulnerabilities.
- ZATS communication subsystems will contain no security vulnerabilities.
- ZATS will have sufficient countermeasures, which will be properly installed and maintained.

Example ZATS Safety Vulnerability Rqmts

Prevention Requirements:

- **Defects:** ZATS shall not contain any safety vulnerabilities in the form of defects in safety-related hardware or software including safeguards.

Detection Requirements:

- **Preventative Maintenance:** ZATS shall provide a means for maintainers to record when safety-critical preventative maintenance is performed.

Reaction Requirements:

- **Preventative Maintenance:** ZATS shall notify its maintainers when safety-critical preventative maintenance has not been performed by its scheduled due date.

Example ZATS Security Vulnerability Rqmts

Prevention Requirements:

- **Defects:** ZATS shall not contain any security vulnerabilities in the form of defects in security-related hardware or software including countermeasures.
- **Bank Card Information:** ZATS shall encrypt passenger sensitive bank card information.
- **System Startup:** On system startup, ZATS shall ensure that its countermeasures are properly configured.
- **Obsolete Virus Detection:** ZATS shall automatically update its virus definitions on a daily basis.

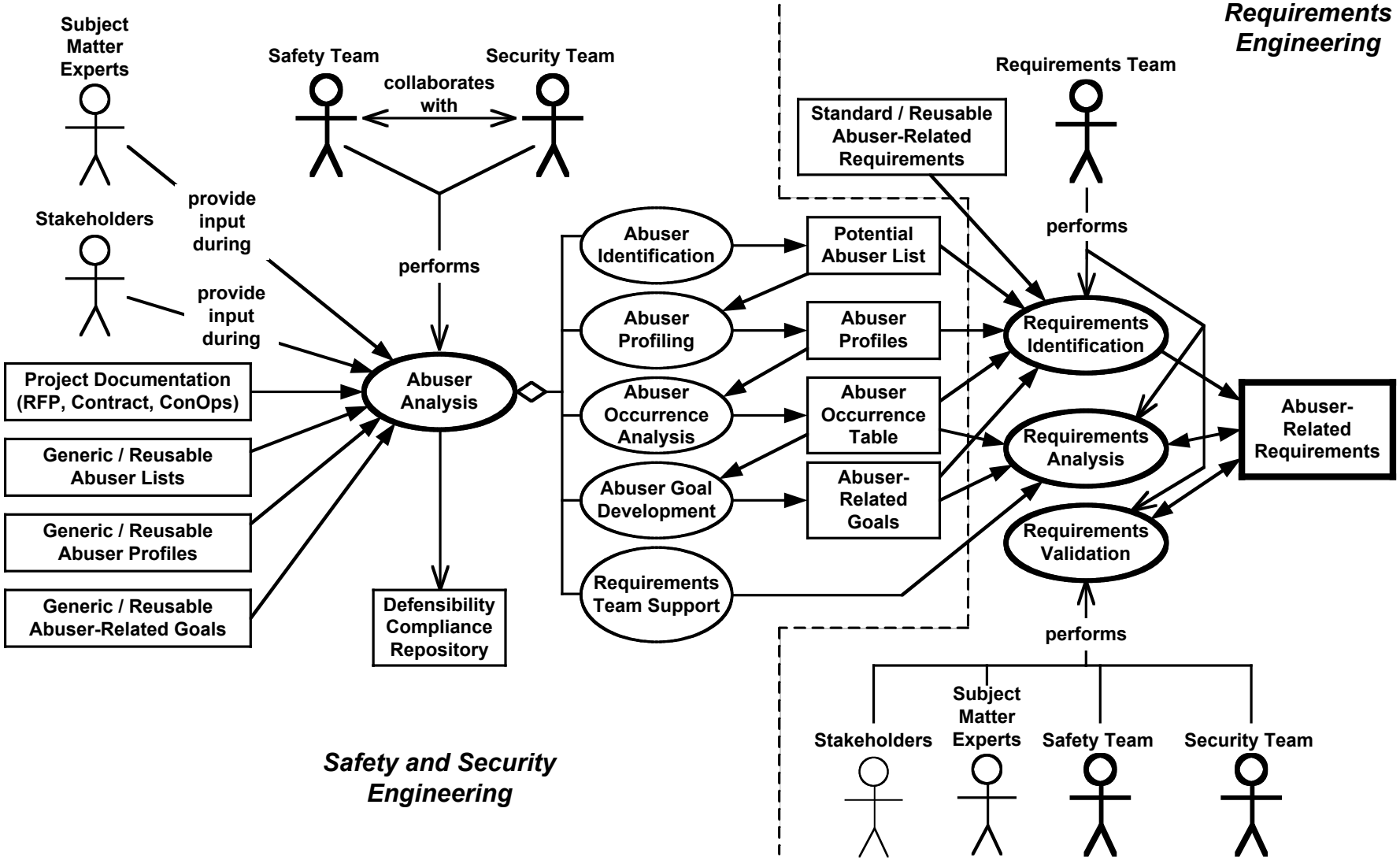
Detection Requirements:

- **Obsolete Virus Detection:** ZATS shall contain technical means to detect if its virus definitions are obsolete.

Reaction Requirements:

- **Obsolete Virus Detection:** If ZATS detects that its virus definitions are obsolete, then it shall automatically update its virus definitions.

Abuser Analysis



Example ZATS Abusers

Non-Malicious Abusers (Safety):

- Human Abuser (e.g., Developer, Maintainer, Operator, Passenger)
- External Systems (e.g., Communications Network, Electrical Power Grid)
- Natural Environment (e.g., River or Weather)

Malicious Abusers (Security):

- Attackers (e.g., Arsonists, Crackers, Disgruntled Current or Former Employees, Terrorists, Thieves)
- Malware:
  - Virus, Trojan horse, and Worm
  - Software, Hardware, and System

Abuser Profiles

Profiles are Highly Reusable

Profiles more commonly used for Attackers than Non-Malicious Abusers.

Attacker Profiles are defined largely in terms of:

- *Means* including Tools, Training, Expertise, Support, and Time
- *Motive* including both Desired Harm and Risk Aversion
- *Opportunity* including Access to System and Valuable Assets

Example ZATS Abuser Goals

Safety Abuser Goals:

- ZATS will provide operators and maintainers with on-line help and training.

Security Abuser Goals:

- To the extent practical, ZATS will prevent attackers from having access to security-related ZATS components and valuable assets.

Example ZATS Abuser Requirements

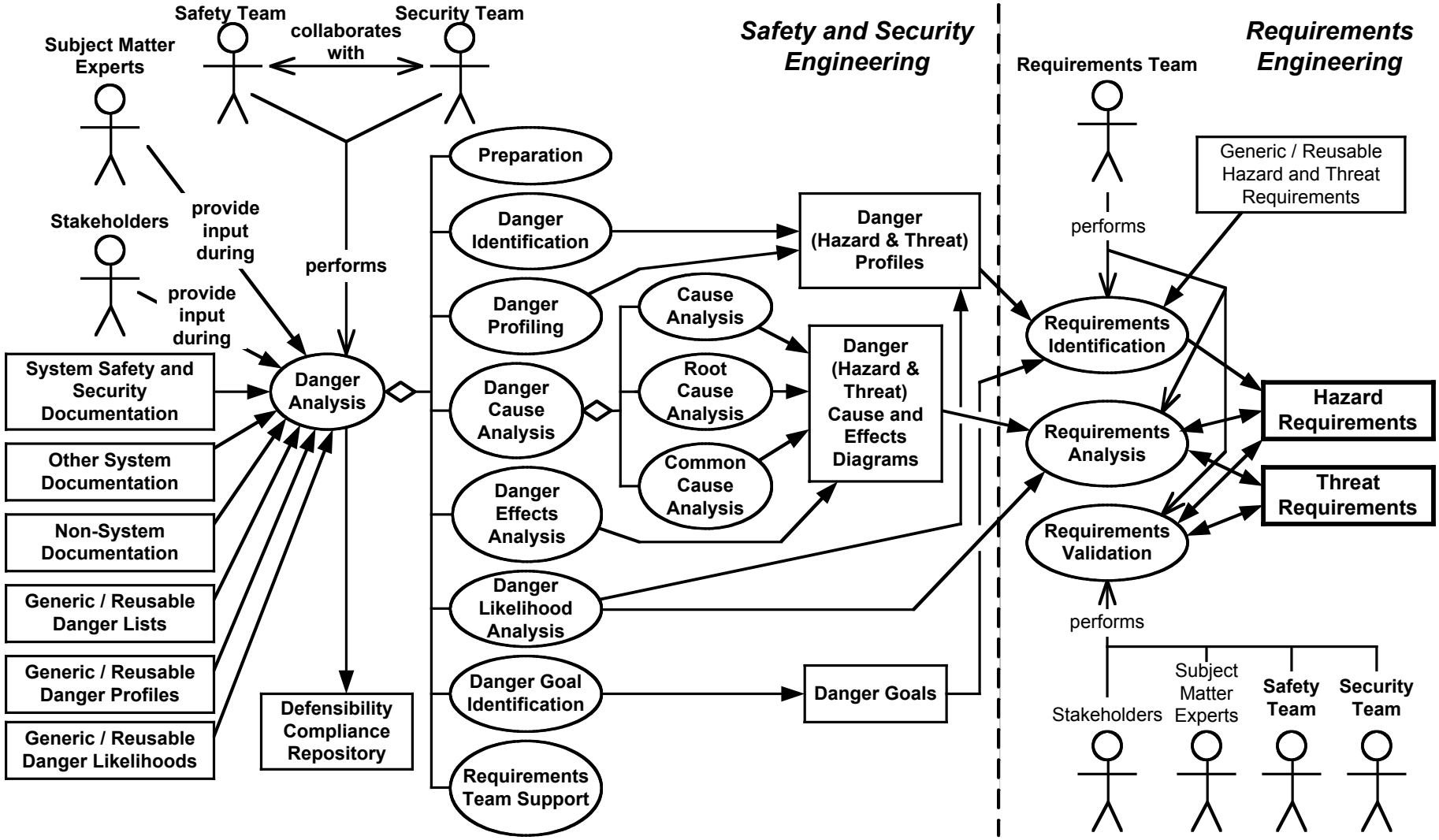
Safety Abuser Requirements:

- Safety-Critical functions shall require multiple operator/maintainer inputs
- Access Control

Security Abuser Requirements:

- Access Control

Danger Analysis



Hazard Analysis (Safety)$_1$

Hazard analysis usually implies the analysis of assets, harm, incidents, hazards, and risks.

Hazard analysis often occurs multiple times before various milestones:

- Preliminary Hazard Analysis (PHA)
- System Hazard Analysis (SHA)

Hazard analysis should probably be performed continuously.

Hazard Analysis (Safety)$_2$

Traditional hazard analysis techniques:

- Come from reliability analysis
- Concentrate on failure analysis
- Do not address all safety concerns

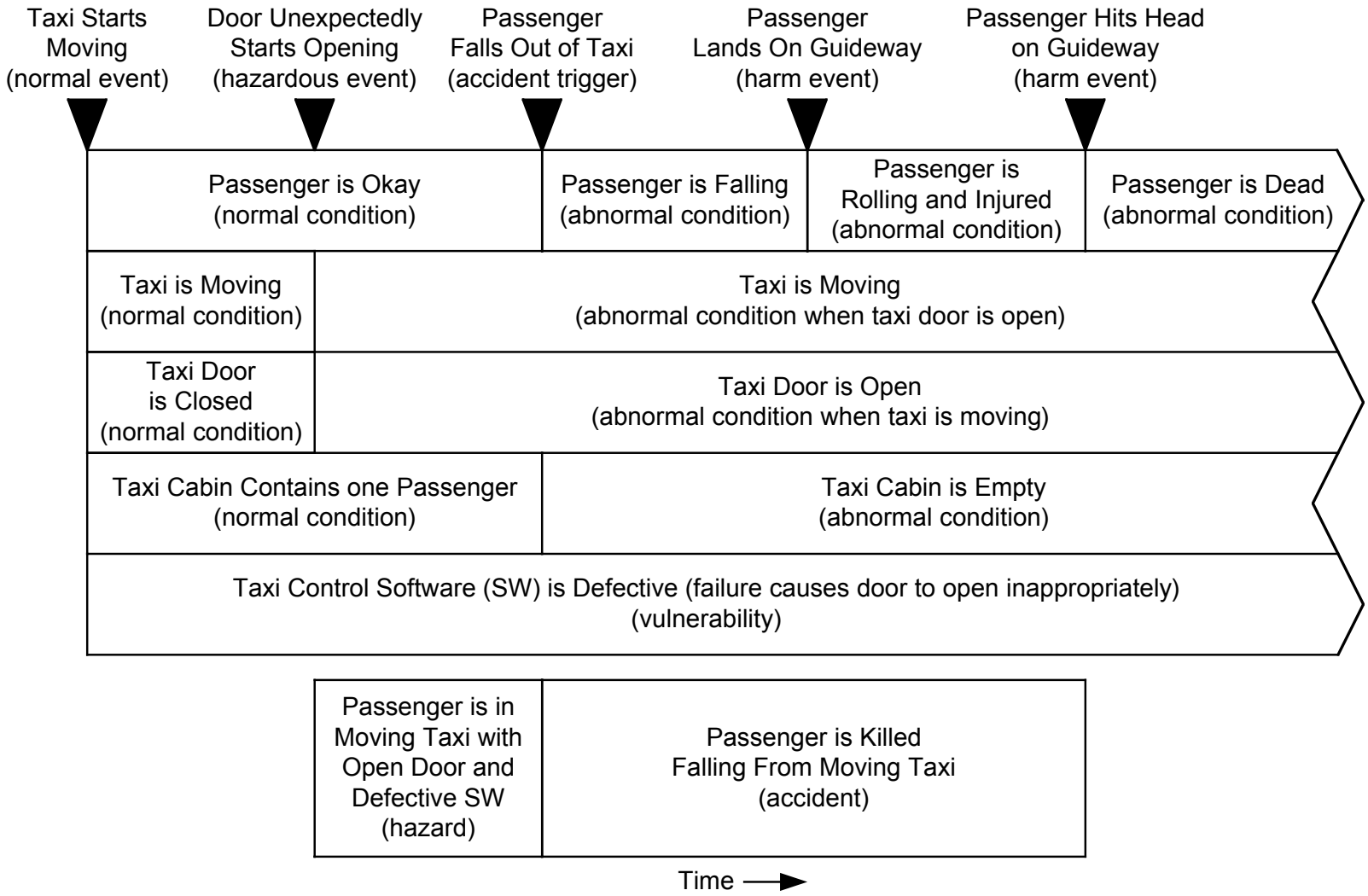
Safety and reliability are not the same:

- Unreliable system that is safe (does nothing)
- Safe system that is unreliable (failures do not cause accidents)

Techniques include:

- Event Tree Analysis (ETA)
- Fault Tree Analysis (FTA)
- Hazard Cause and Effect Analysis (HCEA)
- Failure Mode Effects Criticality Analysis (FMECA)

Example ZATS Hazard, Events, Harm, and Assets



Fault Tree Analysis (FTA)

Develop fault trees
(deductive, backwards-search, decision trees) to identify *causes of failures*.

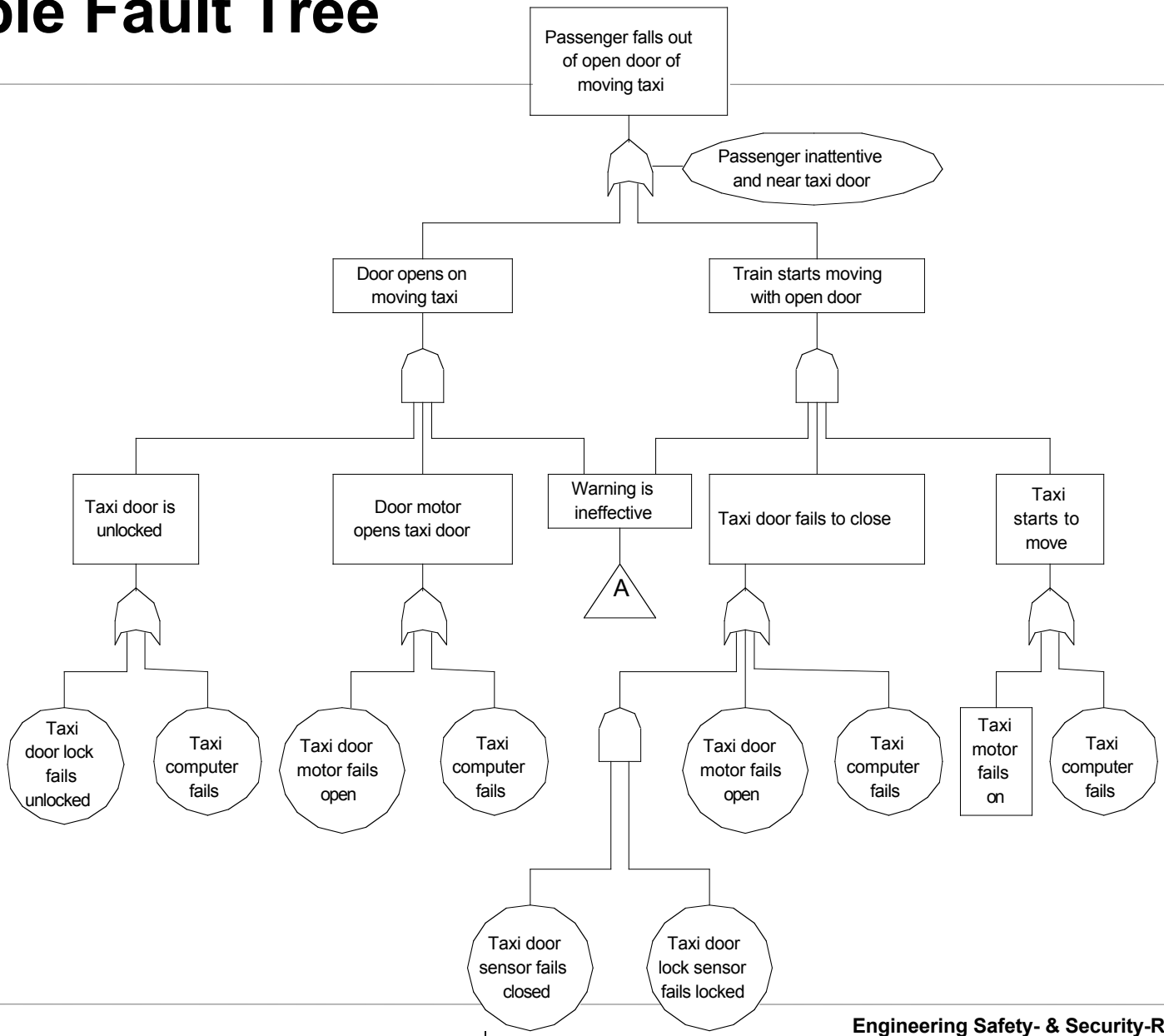
Advantages:

- Long history (1962) of successful use (system/hardware reliability)
- Good documentation
- Well known (in reliability engineering community)
- Good tool support
- Can support quantitative analysis (often impractical for SW)
- Can be used to (indirectly) identify hazards, common causes of safety events, safeguards, and associated requirements

Disadvantages:

- System architecture needed
- Only events, not states (e.g., hazards)
- Non-intuitive symbology that is inconsistent with event trees
- Very expensive and time consuming to produce
- Requires significant analyst expertise
- Ignores system mode

Example Fault Tree



Event Tree Analysis (ETA)

Develop Event Trees (inductive, forwards-search, decision trees) to identify *consequences of failures*.

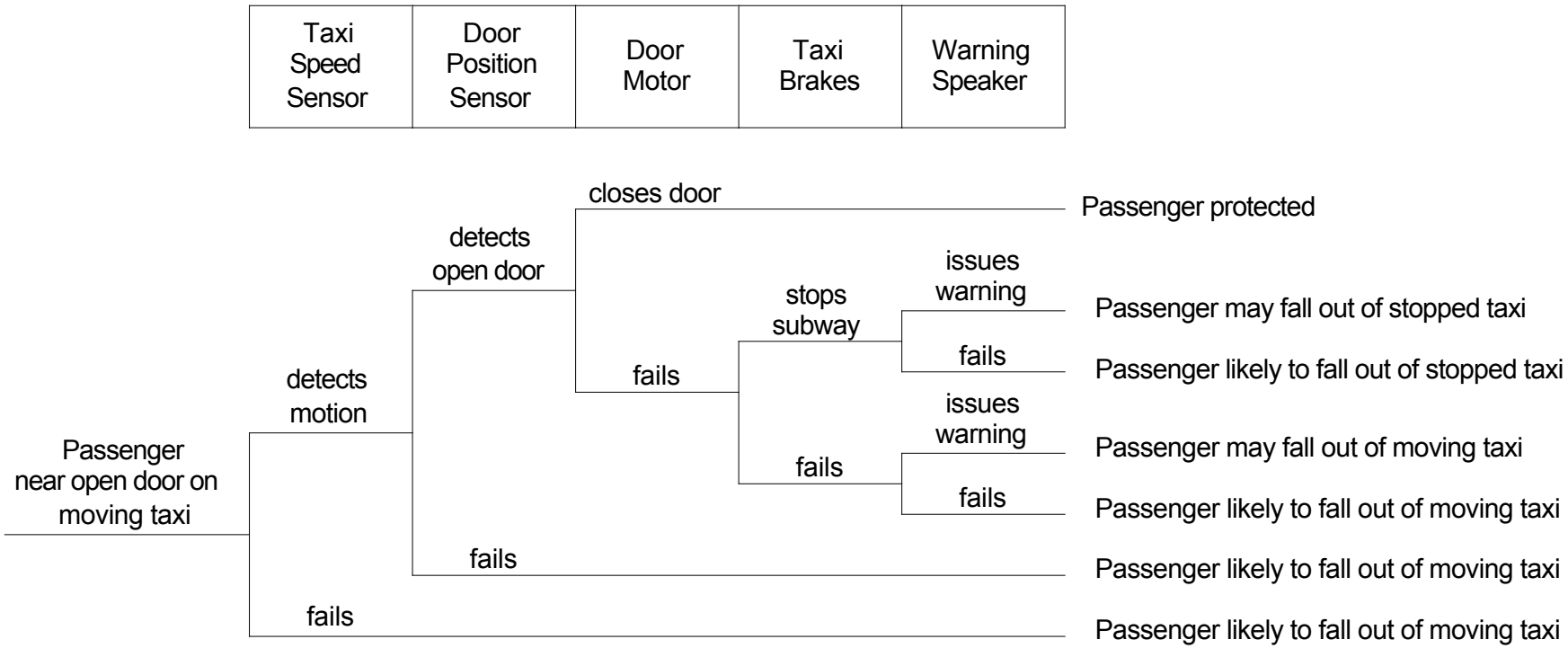
Advantages:

- Long history (1962) of successful use (system/hardware reliability)
- Good documentation
- Well known (in reliability engineering community)
- Good tool support
- Can be used to (indirectly) identify safety events, accidents, and associated requirements

Disadvantages:

- System architecture needed
- Only events, not states (e.g., hazards)
- Non-intuitive symbology that is inconsistent with fault trees
- Very expensive and time consuming to produce
- Not all failures lead to safety events, accidents, and incidents
- Ignores system mode

Example Event Tree



Hazard Cause and Effect Analysis (HCEA)

Develop cause/effect graphs (deductive and inductive, backwards and forwards search, decision trees) to identify *causes* and *consequences* of *safety events and conditions*.

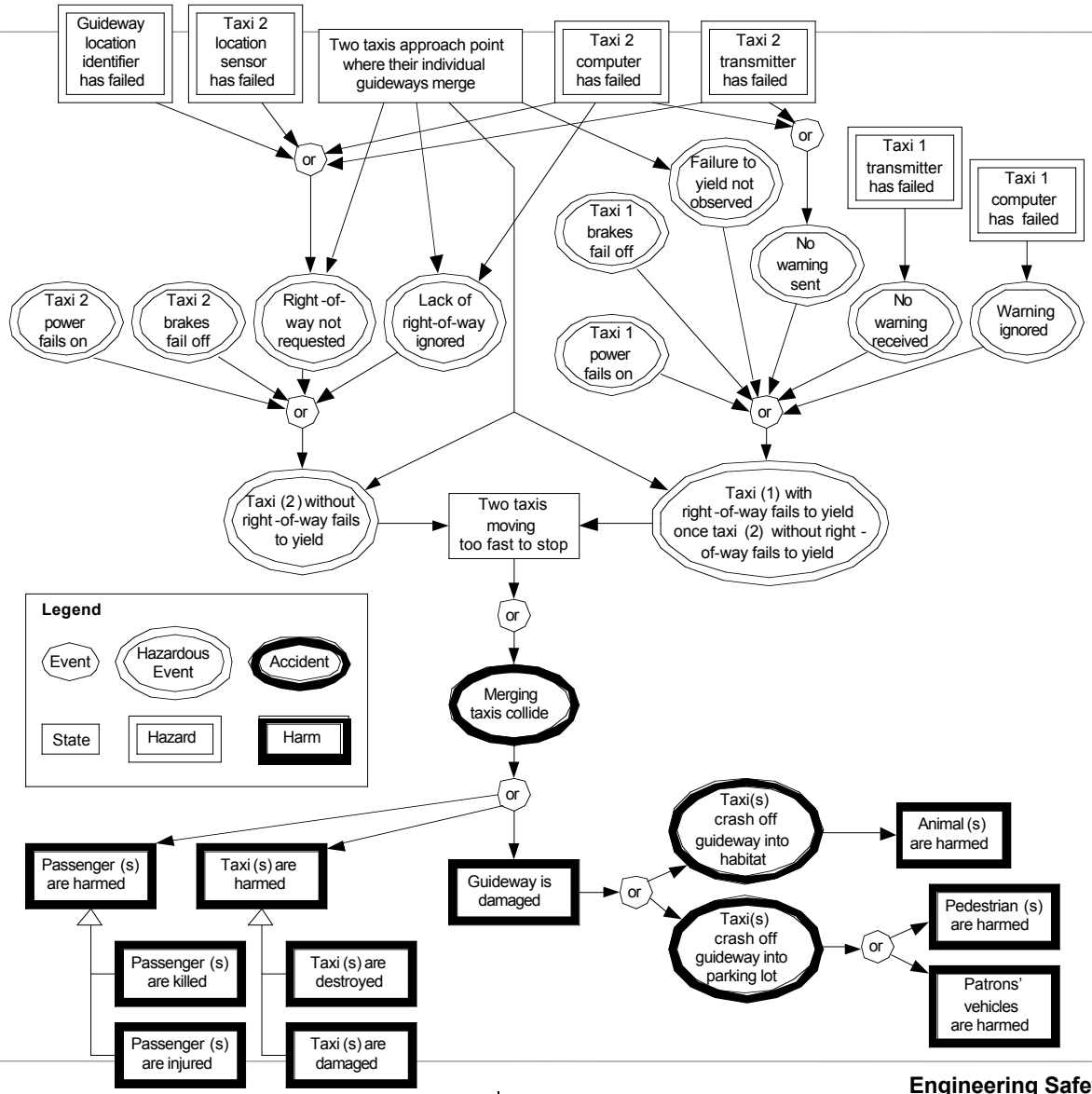
Advantages:

- Designed for Safety Analysis
- Emphasize both Events and States (hazards)
- Use single, compatible notation
- Identifies safety events, accidents, incidents, safety conditions, and associated requirements

Disadvantages:

- Short history
- Not much documentation
- Not well known
- No tool support
- Very expensive and time consuming to produce
- Ignores system modes

Example Cause and Effect Tree



Comparison of Graphical Techniques

Comparison of Techniques	Event Tree Analysis (ETA)	Fault Tree Analysis (FTA)	Hazard Cause Effect Analysis (HCEA)
Analysis Type	Inductive Analysis	Deductive Analysis	Inductive and Deductive Analysis
Graph Type	Event Trees	Fault Trees	Cause and Effect Graphs
Search Direction	Forwards Search	Backwards Search	Forwards and Backwards Search
Scope	Cause of Failure Events (Causes of Accidents)	Effects of Failure Events (Effects of Hazards)	Causes and Effects of Safety Events and Conditions
Use	Identify Hazards	Identify Accidents	Identify Hazards, Accidents, and Incidents
Domain	Reliability (Safety) Analysis	Reliability (Safety) Analysis	Safety Analysis

Failure Modes Effects Criticality Analysis (FMECA)

FMECA stores relevant *reliability* information in tabular form:

- Architecture component that can fail
- Failure Mode (of component)
- Failure Cause
- Possible Effects of Failure
- Effect Severity (i.e., harm severity)
- Failure Probability
- Criticality (risk)
- Reliability Controls

May need to be Restricted To or Developed For Safety and Security

Example ZATS FMECA Table

Component that fail	Failure Mode	Failure Cause	Failure Effect	Failure Severity	Failure Likelihood	Criticality (Risk)	Safety Controls
Accelerometer (Taxi sensor)	No data, Bad data (0, last value, maximum value)	Hardware failure, loss of electrical power, wiring fails	Excessive acceleration or deceleration causing passenger injury	Minor	Low	Moderate	Hardware redundancy, High-reliability COTS component, SW fault tolerance
Computer Hardware (Taxi)	Loss of function (complete or intermittent), bad data	CPU, electrical power (loss or spike), hard drive, motherboard, or RAM failure, high temperature	Taxi not controllable (e.g., braking, power, steering), collision between taxis, collision with guideway, unexpected or emergency braking	Severe	Moderate	Critical	Hardware redundancy, High-reliability COTS components, temperature sensor, SW fault tolerance
Computer Software (Taxi)	Loss of function (complete or intermittent), incorrect function, bad timing of function	CPU, electrical power (loss or spike), hard drive, motherboard, or RAM failure, high temperature	Taxi not controllable (e.g., braking, power, steering), collision between taxis, collision with guideway, unexpected or emergency braking	Severe	High	Critical	SEAL 1 applied (e.g., SW fault tolerance, real-time operating system, safe language subset, formal specification of core functions, etc.)

Example ZATS Hazards

Taxi Moving With Open Door Hazard:

- Existence of **Vulnerable Assets**: Passenger in Taxi
- Existence of **Vulnerabilities**: Lack of or Defective Door Look, Door Motor, Door Sensor, Speed Sensor, Taxi Processor, and/or Associated Software
- Existence of **Non-malicious Abusers**: Inattentive or Careless Passengers, Maintainers, and/or Software Developers
- Existence of **Other Conditions**: Taxi Moving and Taxi Door Open

Taxi Station Fire Hazard:

- Existence of **Vulnerable Assets**: Taxi Station and Passengers in Taxi Station
- Existence of **Vulnerabilities**: Lack of or Defective Fire Detection and Suppression System (e.g., Sensors, Water Sprayers, Alarm, Communication Equipment, Processor, and/or Associated Software)
- Existence of **Non-malicious Abusers** : Inattentive or Careless Maintainers and/or Software Developers
- Existence of **Other Conditions**: Flammable Materials

Example ZATS Threats

Threat to Bank Card Information:

- Existence of **Vulnerable Assets**: Bank Card Information
- Existence of **Vulnerabilities**: Lack of or Defective Encryption/Decryption Subsystem,
- Existence of **Malicious Abusers**: Attackers (Cyber-thieves) and Malware
- Existence of **Other Conditions**: Connection to Bank Card Processing Gateway over the Internet or Dedicated Line

Threat of Virus Infection:

- Existence of **Vulnerable** Assets: Processors
- Existence of **Vulnerabilities**: Lack of or Inadequate Maintenance of Virus Protection Software, Connection of Servers to the Internet, Use of Windows OS, …
- Existence of **Malicious Abusers**: Attackers (Script Kiddies) and Malware (Viruses and Worms)
- Existence of **Other Conditions**: Existence of Updates to Taxi and Taxi Station Software

Example ZATS Danger Goals

Safety Hazard Goals:

- ZATS will prevent passengers from falling out of moving taxis.
- ZATS will prevent taxi station fires.

Security Threat Goals:

- ZATS will protect passenger bank card data from cyber-thieves.
- ZATS will protect itself from infection by computer viruses, worms, etc.

Example ZATS Hazard Requirements

Moving With Open Door Hazard Requirements:

- ZATS taxis shall not move when their doors are open* at a rate of more than X per trip.
- ZATS taxis shall not open their doors when moving at a rate of more than X per trip.

Fire Hazard Requirements:

- ZATS shall detect smoke above X ppm in the taxi stations within 2 seconds at least 99.9% of the time.
- ZATS shall detect temperatures above X° C in the taxi stations within 2 seconds at least 99% of the time.
- Upon detection of smoke or excess temperature in the taxi stations, ZATS shall begin fire suppression within 1 second and notify the ZATS operator and the Fire Department within 5 seconds at least 99.9% of the time.

* Note that the terms moving and open doors must be unambiguously defined.

Example ZATS Threat Requirements

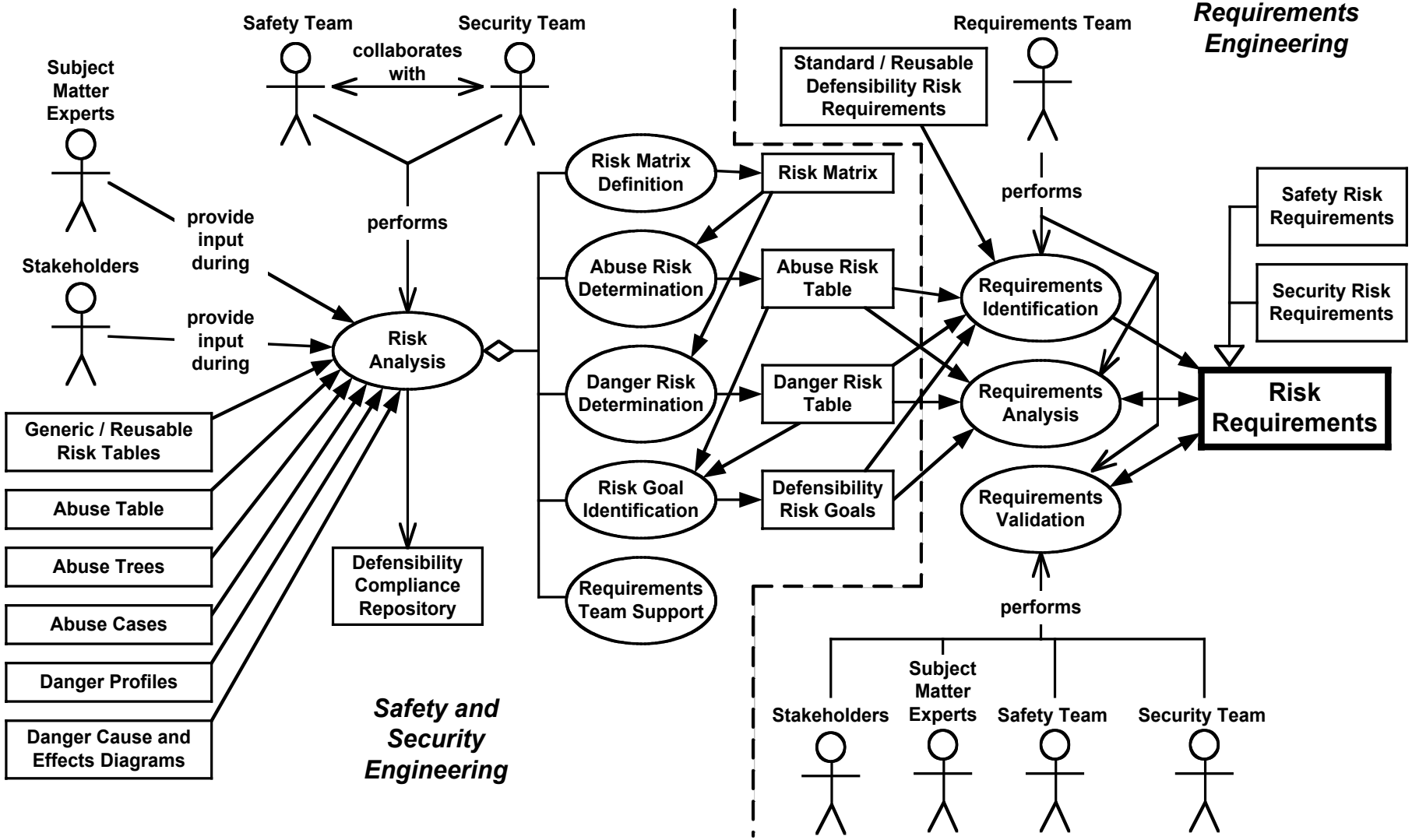
Threat to Bank Card Information Requirements:

- ZATS shall use sufficient technical means (e.g., encryption, digital signatures, and hash codes) to protect the confidentiality, integrity, and nonrepudiation of *transmitted* passenger bank card information from attackers of type W having profiles of type X for a minimum duration of time Y with a confidence of Z.
- ZATS shall use sufficient technical means (e.g., encryption, digital signatures, and hash codes) to protect the confidentiality and integrity of *stored* passenger bank card information from attackers of type W having profiles of type X for a minimum duration of time Y with a confidence of Z.

Virus Threat Requirements:

- ZATS shall use sufficient technical means (e.g., virus suppression subsystem, manual/automatic virus definitions update procedures) to protect the ZATS servers from infection by known viruses, worms, Trojans, etc.

Defensibility Risk Analysis



Example Safety Risk Matrix

Safety Risk Matrix defines safety risk (and associated SAL) as a function of:

- Harm Severity
- Frequency of *Abuse* Occurrence or *Danger* Existence

Safety Risks / Safety Assurance Levels (SALs)					
	Frequency of Abuse / Danger Occurrence				
Harm Severity	**Frequent**	**Probable**	**Occasional**	**Remote**	**Implausible**
Catastrophic	Intolerable	Intolerable	Intolerable	High	Medium
Critical	Intolerable	Intolerable	High	Medium	Medium
Major	High	High	Medium	Medium	Low
Minor	High	Medium	Low	Acceptable	Acceptable
Negligible	Medium	Low	Acceptable	Acceptable	Acceptable

Example ZATS Defensibility Risk Goals

Safety Risk Goals:

- ZATS will not have any intolerable safety risks.

Security Risk Goals:

- ZATS will not have any intolerable security risks.

Example ZATS Defensibility Risk Requirements

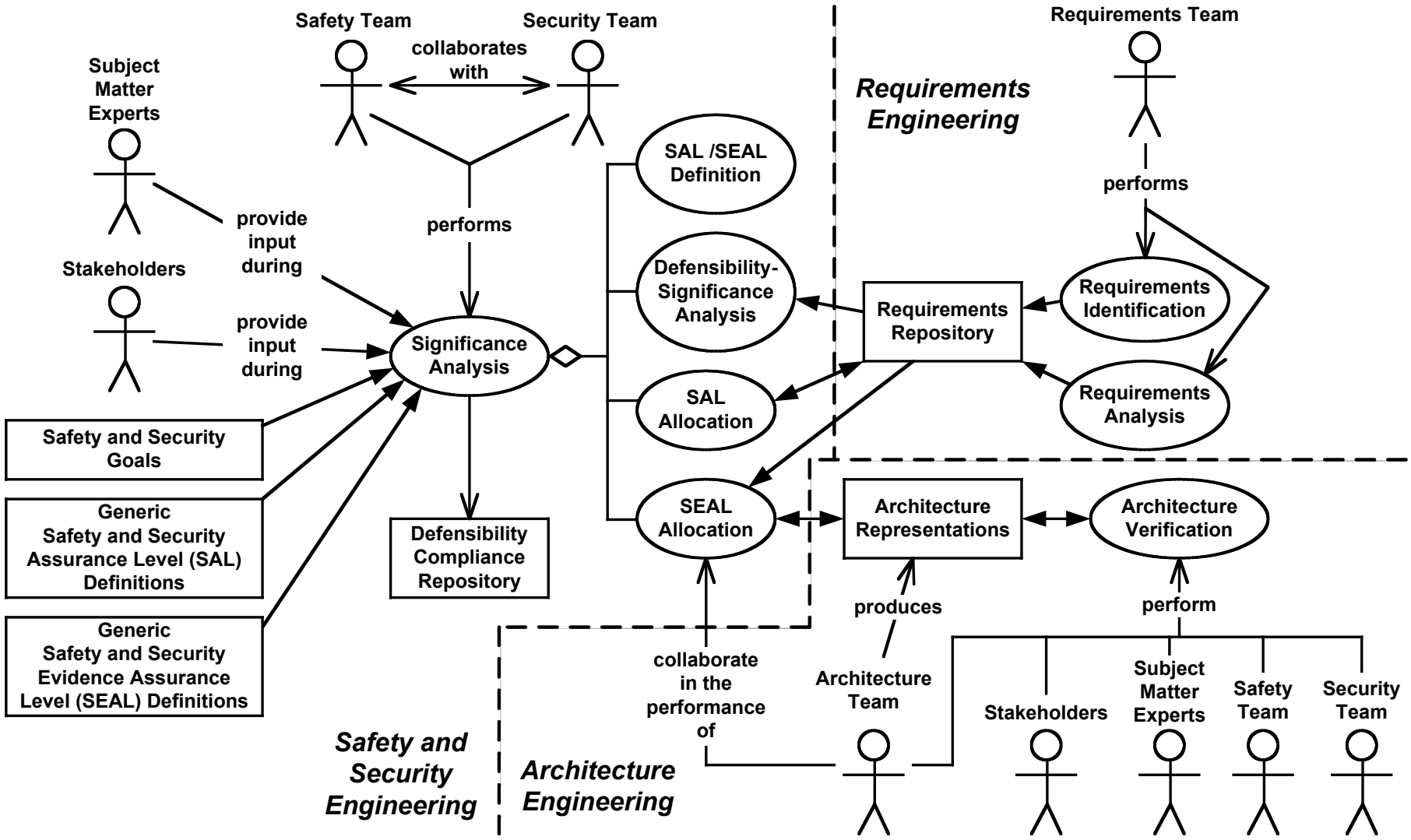
Safety Risk Requirements:

- ZATS shall not have any intolerable safety risks.
- Any ZATS high and medium residual safety risks must be officially accepted by the Metropolitan Zoo Authority and the People Mover Incorporated Executive Management.
- Any ZATS low residual safety risks must be officially accepted by the ZATS program safety team and ZATS PMI program manager.

Security Risk Requirements:

- ZATS shall not have any intolerable security risks.
- Any ZATS high and medium residual security risks must be officially accepted by the Metropolitan Zoo Authority and the People Mover Incorporated Executive Management.
- Any ZATS low residual security risks must be officially accepted by the ZATS program security team and ZATS PMI program manager.

Defensibility Significance Analysis



Safety/Security Assurance Levels (SALs)

Safety/Security Assurance Levels (SALs) are categories of requirements based on their associated safety/security risk level.

SALs can be determined for:

- Individual requirements.
- Groups of related requirements (e.g., features or functions).

SALs should be appropriately, clearly, and unambiguously defined.

Another Example of Safety/Security Assurance Levels (SALs)

Intolerable:

The risk associated with the requirement(s) is totally unacceptable to the major stakeholders. The requirement(s) *must* therefore be deleted or modified to lower the associated risk.

Undesirable:

The risk associated with the requirement(s) is so high that major (e.g., architecture, design, implementation, and testing) steps should be taken to lower the risk (e.g., risk mitigation and risk transfer) to lower the risk.

As Low As Reasonably Practical (ALARP):

Reasonable practical steps should be taken to lower the risk associated with the requirement(s).

Acceptable:

The risk associated with the requirement(s) is acceptable to the major stakeholders and no additional effort must be taken to lower it.

Example ZATS Safety-Significant Requirement Categories

Controlling Doors (Opening, Closing, Locking)

- Passengers / Property May Fall Out of Taxi

Accelerating and Decelerating (Power Braking System)

- Taxis May Collide

Merging Traffic

- Taxis May Collide

Example ZATS Security-Significant Requirement Categories

Pay for Trips with Bank Cards and Cash

- Confidential Bank Card Information Compromised

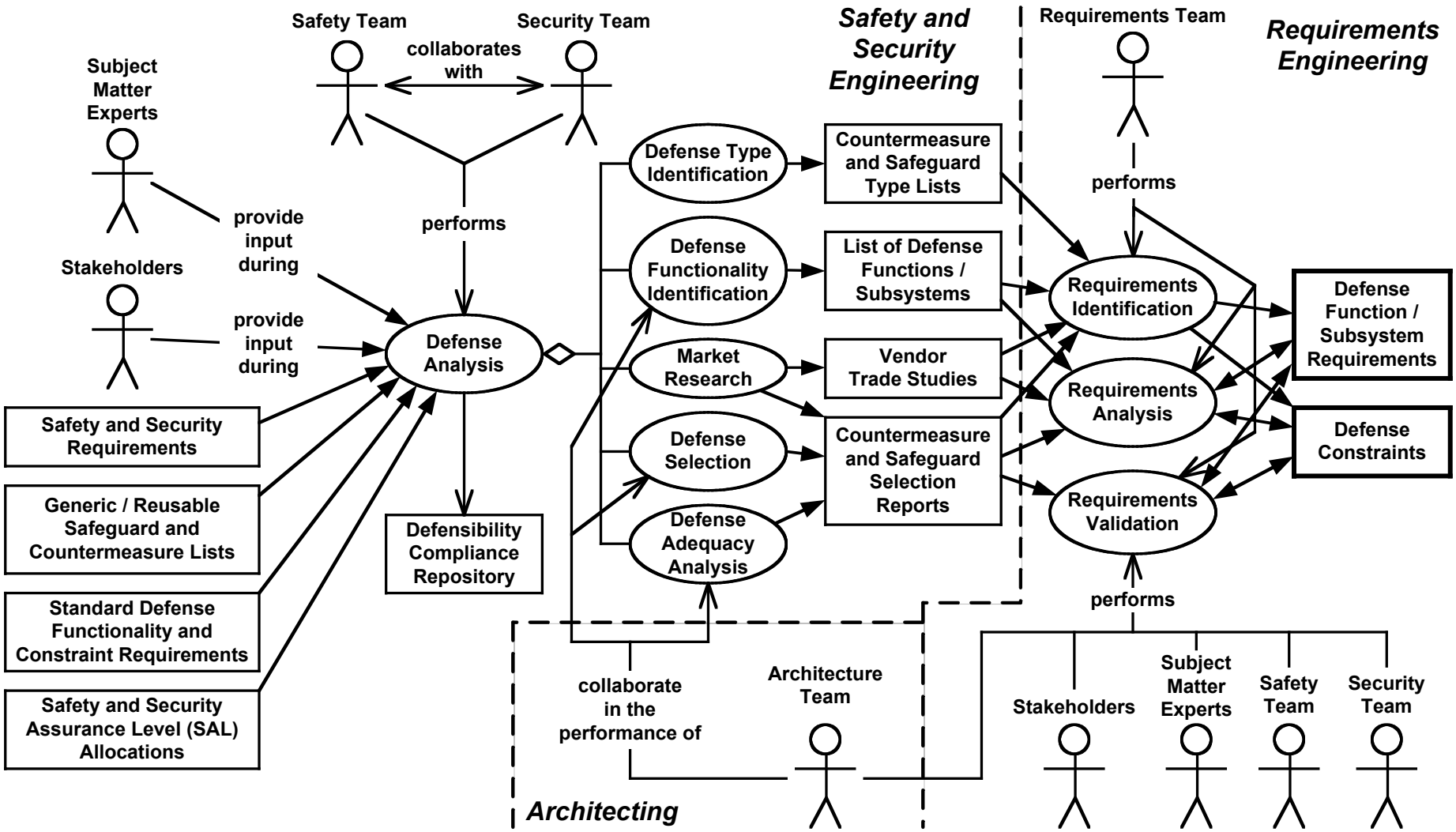
Provide Free Travel to Zoo Employees and Premier Zoo Patrons

- Confidential Information Compromised

Passengers Embarking/Disembarking in Parking Lots

- Mugging in Deserted Parking Lots
- Arson of Taxi Station outside of Zoo Fence

Defense Analysis



Example ZATS Defensibility Functions / Subsystems

Safety Functions and Subsystems:

- Emergency Communication System
- Fire Detection and Suppression
- Taxi Safety Subsystem
- Taxi Station Door Subsystems
- Video Surveillance

Security Functions and Subsystems:

- Access Control
- Emergency Communication
- Encryption/Decryption
- Intrusion Detection
- Security Audit
- Video Surveillance
- Virus Detection

Example ZATS Safety Constraints

"When the vehicle is stopped in a station with the doors open for boarding, the horizontal gap between the station platform and the vehicle door threshold shall be no greater than 25 mm (1.0 in.) and the height of the vehicle floor shall be within plus/minus 12 mm (0.5 in.) of the platform height under all normal static load conditions…"

Automated People Mover Standards – Part 2: Vehicles, Propulsion, and Braking (ASCE 21-98)

ZATS shall use pre-stressed reinforced concrete guideways able to support 150% max. expected loading.

ZATS guideways shall be elevated to provide good visibility, to separate patrons from the animals, and to eliminate the possibility of collision between taxis and patrons' vehicles in the parking lots.

Oils and hydraulic fluids shall be *flame retardant*, except as required for *normal lubrication*.

Note need to define flame retardant and normal lubrication.

ZATS software shall be programmed in a safe subset of C++.

Example ZATS Security Constraints

"Servers shall be protected by firewalls."

"Users shall be identified and authenticated by textual user IDs and associated pass phrases."

"Sensitive data shall be protected by use of a COTS public key encryption/decryption product."

"Malware infection shall be prevented by the use of a COTS antivirus product."

Conclusion:

Process Improvement Recommendations

You Are Here

Three Disciplines

Challenges

Requirements Engineering Overview

Safety and Security Engineering Overview

Types of Safety- and Security-related Requirements

Common Consistent Collaborative Method

Conclusion _

Conclusion$_1$

Engineering safety-significant requirements requires *appropriate*:

- Concepts
- Methods
- Techniques
- Tools
- Expertise

These must come from*:*

- Requirements Engineering (Safety- and Security-related Requirements)
- Safety Engineering (Analysis and Safety Goals)
- Security Engineering (Analysis and Security Goals)

Conclusion$_2$

There are four types of Safety- and Security-related Requirements:

- Safety and Security Quality Requirements
- Safety- and Security-Significant Requirements
- Safety and Security Function/Subsystem Requirements
- Safety and Security Constraints

Different Types of Safety- and Security-related Requirements often have different Structures.

These different Types of Requirements need to be identified, analyzed, and specified differently.

Conclusion$_3$

Processes for Requirements Engineering, Safety Engineering, and Security Engineering need to be:

- Properly interwoven.
- Consistent with each other.
- Performed collaboratively and in parallel (i.e., overlapping in time).

Final Thoughts

Look for my upcoming book by the same title to be published by Auerbach.

The slides for this tutorial will be put onto the SEI Website in the next 2 weeks, probably on the ASP Publications webpage:

www.sei.cmu.edu/programs/acquisition-support/presentations.html

Questions?

Donald Firesmith
Acquisition Support Program (ASP)
Software Engineering Institute (SEI)
Pittsburgh, Pennsylvania, USA 15213
412-216-0658 (cell)
dgf@sei.cmu.edu

Software Engineering Institute | Carnegie Mellon